uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Telemática
2017-2018

Trabajo Fin de Grado

Estudio de la plataforma de orquestación NFV OSM

Alberto Barrado Vicente

Tutor: Carlos Jesús Bernardos Cano

Director: Pablo Serrano Yáñez-Mingot

Leganés, 4 de octubre de 2018

Resumen

La industria de las telecomunicaciones está en constante evolución y en los últimos años se ha encontrado ante la obligación de utilizar arquitecturas más flexibles debido a las nuevas exigencias del mercado. Los diferentes proveedores de servicios se dieron cuenta que era necesario desarrollar las infraestructuras, puesto que los requisitos requeridos a corto plazo iban a exceder el límite de las redes tradicionales, sobre todo en volumen de información y velocidad. Como consecuencia de dicha limitación surge Network Function Virtualization (NFV), un marco de referencia relacionado con la arquitectura de las redes, orientado a virtualizar diferentes elementos dentro de las mismas.

La arquitectura de las redes tradicionales es estática, debido a que el hardware y el software están estrechamente integrados. Por el contrario, NFV permite que el software se utilice en plataformas diferentes, permitiendo varios puntos de gestión de la red, los cuales se pretende que sean estándar y compatibles con soluciones de varios fabricantes. De esta manera se consigue sustituir dispositivos hardware complejos y especializados por sistemas equivalentes que se ejecutan sobre servidores genéricos. Se pretende pasar de soluciones aisladas sin posibilidad de compartir recursos, a soluciones basadas en plataformas compartidas, que ofrecen libertad a la hora de seleccionar fabricantes y flexibilidad.

Para conseguir una solución NFV hay que implementar primero su arquitectura, en la que existen tres componentes principales: Network Functions Virtualization Infrastructure (NFVI), Virtualized Network Functions (VNFs) y Management and Orchestration (MANO). Aunque todos los componentes de la arquitectura son igual de importantes, este Trabajo Fin de Grado (TFG) se centra en el estudio del bloque MANO. Esto se debe a que Open Source MANO (OSM), el objeto de estudio de este proyecto, es una implementación de este componente.

MANO es un bloque separado dentro de la arquitectura NFV, que interactúa tanto con la NFVI como con las VNFs. Es el responsable del control de todas las entidades dentro de la arquitectura NFV. Por tanto, debe conocer el uso, el estado, las estadísticas y el resto de los parámetros asociados a todos los elementos implicados en la arquitectura. A su vez el bloque MANO está formado por tres componentes: VNF Manager (VNFM), NFV Orchestrator (NFVO) y Virtual Infraestructure Manager (VIM).

Una vez introducidos los elementos más relevantes de la tecnología NFV, se puede hablar de Open Source MANO, la plataforma principal del estudio. OSM nació como una iniciativa del European Telecommunication Standards Institute (ETSI) para desarrollar una pila de software libre de gestión y orquestación que siga los estándares definidos para la tecnología NFV. Fue creado por un conjunto de empresas, principalmente proveedores de servicio y vendedores de soluciones. Entre sus miembros fundadores se encuentran Telefónica, BT, RIFT.io, Mirantis, Telekom Austria Group, Intel, Telenor y Canonical. La definición más acertada para calificar a OSM es la de una

comunidad de código abierto, cuyo objetivo es ofrecer una pila de software MANO de calidad para producción de tecnología NFV. Pretende ser compatible con cualquier VNF, accesible para todos los usuarios e independiente del módulo VIM. El código de Open Source MANO está desarrollado de acuerdo a los procedimientos establecidos para el código abierto. En la plataforma de desarrollo de OSM, gestionada por la ETSI, se encuentra la información más reciente, con el fin de que cualquier usuario interesado pueda acceder a ella. OSM está en constante evolución y desde su nacimiento, a finales de 2016, ha sufrido varias modificaciones, con el fin de adaptarse a los avances del sector y mejorar su funcionalidad. Cada 6 meses aproximadamente sale al mercado una nueva versión que incluye cambios notables respecto a la anterior. La versión más actual es la número cuatro, más conocida como OSM FOUR, que salió a principios de mayo de 2018.

OSM tiene una arquitectura modular que ha ido evolucionando constantemente para adoptar los principios de diseño de la nube. Cada uno de los módulos realiza funciones concretas e independientes, que en conjunto ofrecen la funcionalidad completa de una plataforma de orquestación NFV. Cada módulo de la arquitectura está instalado sobre un contenedor, el cual es un conjunto de procesos que están separados del resto del sistema. Los contenedores son como "cajas" aisladas que disponen únicamente de los recursos imprescindibles para poder ejecutar una determinada aplicación o programa. Dentro de la arquitectura de OSM existen varios módulos, que son explicados más detenidamente en el estudio, sin embargo, hay uno que merece ser comentado antes que los demás, no por ser el componente más importante de la arquitectura, sino por la gran importancia que ha adquirido tras la salida de la última versión de OSM. Se trata del módulo Monitoring (MON), que era únicamente un componente experimental con funcionalidades muy limitadas, sin embargo, ha adquirido un mayor protagonismo, pasando a forma parte de la arquitectura principal de OSM. Existen muchas herramientas de monitorización en el mercado y la finalidad del módulo MON no es competir con ellas, sino actuar como intermediario entre dichas herramientas y el orquestador. Por lo tanto, su función es integrar una herramienta de monitorización concreta, elegida según las necesidades del proyecto, con la arquitectura de OSM.

Otro elemento de especial relevancia es el módulo VIM, el cual es un elemento imprescindible dentro de una arquitectura MANO, sin embargo, OSM no lo implementa. La razón de no incluirlo es no imponer un VIM concreto a la hora de trabajar junto a su arquitectura, permitiendo elegir cual es la plataforma que se adecúa más a cada proyecto. Existen múltiples alternativas, sin embargo, el grupo de trabajo de OSM únicamente recomienda las cuatro que han sido probadas y verificadas tras haber sido integradas junto a la plataforma de orquestación, estas son: OpenVIM, VMware vCloud Director, Amazon Web Services y OpenStack; siendo esta última la plataforma elegida para implementar el módulo VIM de este proyecto.

OpenStack es la plataforma Cloud Computing de software libre más importante y que mayor crecimiento ha experimentado en los últimos años. Está diseñada para ofrecer máquinas virtuales e infraestructuras de almacenamiento en la nube. La arquitectura de OpenStack está formada por varios nodos, que llevan a cabo distintas funcionalidades. La

implementación de todos ellos no es obligatoria y depende de las necesidades de cada proyecto. Existen aproximadamente diez tipos de nodos distintos que pueden formar parte de la arquitectura de OpenStack, sin embargo, solo hay dos que deben estar siempre presentes en cualquier implementación: el nodo controller y el nodo compute. Para las funciones que debe llevar a cabo OpenStack en este TFG solo se necesita implementar estos dos nodos recién mencionados. El nodo controller es dónde reside la inteligencia de la arquitectura, ya que se encarga de coordinar el resto de los nodos y en él se instalan la mayoría de los servicios. El nodo compute es dónde se ejecutan las instancias virtuales y se encarga de los servicios mínimos necesarios para administrar dichas instancias. Cada servicio en OpenStack se encarga de una tarea muy concreta y la puesta en común de varios de ellos es lo que permite que funcione la plataforma. Algunos de estos servicios son imprescindibles para su correcto funcionamiento, sin embargo, hay otros que son completamente opcionales, ofreciendo interesantes funcionalidades que pueden ser añadidas dependiendo de las necesidades de cada implementación concreta.

El objetivo del TFG es analizar a fondo OSM, por lo que realizar un estudio meramente teórico es insuficiente, ya que la mejor manera de estudiar la plataforma es analizarla de una manera experimental. A la hora de realizar un estudio práctico hay que realizar el montaje de la arquitectura de OSM, por lo que el procedimiento de instalación de la plataforma tiene gran importancia para el estudio. Por otro lado, es indispensable realizar el montaje de OpenStack ya que para evaluar correctamente a OSM tiene que tener un módulo VIM asociado. Otro objetivo, con un papel más secundario, pero también importante, es crear un TFG autocontenido, es decir, que en la memoria quede documentado todo lo necesario para poder replicar el estudio, sin necesidad de buscar información en documentos ajenos.

Antes de instalar las dos plataformas necesarias para llevar a cabo el proyecto, es importante definir cómo van a estar implementadas sobre el equipo utilizado para realizar el estudio. Ambas se pueden instalar directamente sobre un equipo o en una máquina virtual (VM), siendo esta última la opción elegida por cuestiones de seguridad y portabilidad. Una vez establecido que se van a usar máquinas virtuales para llevar a cabo el proyecto, se debe usar una herramienta de virtualización para poder crearlas. Existen varias alternativas en el mercado por lo que es imprescindible hacer un análisis de cuál es la más recomendable para realizar este estudio, primando el rendimiento y el aspecto económico.

A la hora de instalar OSM existe únicamente un procedimiento, el cual consiste en utilizar un script que permite ejecutar cientos de sentencias de código de manera automática y así evitar a los usuarios tener que hacerlo de manera manual. Por el contrario, existen dos alternativas a la hora de instalar OpenStack: mediante el script DevStack o realizando la instalación manualmente. Aunque lo más cómodo y rápido sea obtener la plataforma mediante la ejecución del script Devstack, se ha optado por hacer la instalación de manera manual. El motivo principal de esta decisión es que para implementar el módulo VIM únicamente son necesarios algunos servicios de OpenStack y eso solo se puede conseguir mediante la instalación manual, por lo que si se utilizara

Devstack se crearían más servicios de los imprescindibles y no se haría un uso eficiente de los recursos hardware, que son muy limitados.

Una vez que todos los componentes estén funcionando en sus respectivas VMs, se puede evaluar sobre la implementación el potencial de la plataforma OSM, usando para ello, una serie de escenarios o Network Services (NS). Desplegar estos NS permite, en primer lugar, verificar la correcta instalación e integración de todos los componentes de la arquitectura implementada y, posteriormente, evaluar el potencial del orquestador, que es uno de los objetivos principales de este estudio.

Se evalúan tres NS, el primero de ellos ha sido proporcionado por OSM y sirve para ser instanciado a modo de ejemplo, ya que es muy sencillo y por lo tanto no permite evaluar adecuadamente al orquestador. Sin embargo, es excelente como toma de contacto con la plataforma, además permite comprobar que la instalación de los módulos de OSM se ha realizado correctamente. Una vez que se ha probado el NS de ejemplo y, por lo tanto, se ha verificado que OSM funciona correctamente, se pueden instanciar NS más complejos, los cuales sí permiten evaluar el potencial de OSM. Estos NS no existen y por lo tanto hay que diseñarlos y crearlos desde cero, lo cual permite comprender mejor su funcionamiento y evaluar servicios con una finalidad interesante. En este caso, ambos NS están enfocados en la recodificación de vídeo, que puede hacerse de muchas maneras, algunos ejemplos son modificar la tasa a la que se transmite el vídeo, cambiar el modo de transmisión, etc. Ambos NS cuentan con dos VNFs, una se encarga de la emisión de vídeo y la otra de modificar dicha transmisión, es decir, recodifica el vídeo. La diferencia entre ambos NS reside en cómo se transmite el vídeo, mientras que, en el caso más sencillo, los usuarios solo pueden visualizar el vídeo a la vez que se está emitiendo, como ocurre con la televisión, en el caso más complejo se puede ver el vídeo en cualquier momento, ya que se transmite mediante un servidor de vídeo bajo demanda, al igual que ocurre con plataformas como YouTube.

Una vez determinado el potencial de la plataforma con los NS se puede realizar una evaluación de OSM, que está centrada en un análisis del rendimiento de la plataforma. Para ello se evalúan los tiempos de despliegue de la propia plataforma, de un NS sencillo y de otro complejo. También se prueban funcionalidades poco comunes que incorpora OSM y lo diferencian del resto de orquestadores, como la inclusión de aplicaciones complementarias en la propia instalación de la plataforma o la funcionalidad VIM-emulator.

Una de estas aplicaciones complementarias es Grafana, la cual permite visualizar gráficamente algunos valores sobre las instancias realizadas en OSM, como pueden ser la utilización de CPU, los paquetes recibidos, etc. La integración con Grafana pretende potenciar el uso del módulo MON, que como ya se comentó anteriormente, ha adquirido un gran protagonismo a raíz del lanzamiento de la nueva versión de OSM. Esto se debe a que MON simplemente proporciona los datos de forma numérica, difíciles de utilizar para las personas, por lo que la integración con Grafana, u otra aplicación similar, ayuda a interpretar de una manera más sencilla los datos obtenidos. Otra de estas aplicaciones es Kibana, que al igual que Grafana, permite visualizar gráficamente algunos datos de OSM,

pero en este caso no se trata de representar los valores medidos en las instancias sino de la visualización de eventos. Kibana proporciona la libertad de seleccionar la forma en que se visualizan los datos, pudiendo elegir: histogramas, gráficos de barras, gráficos circulares y algunos más sofisticados. Su uso permite comprender mejor las estadísticas de OSM e incluso detectar comportamientos anómalos dentro de la plataforma. Por otro lado, VIM-emulator permite no tener que asociar un VIM junto a la arquitectura de OSM ya que se emula dicho módulo en la propia plataforma. Se trata de una función muy útil que permite probar rápidamente una arquitectura MANO completa, sin necesidad de crear un VIM, que es un proceso complejo.

En el estudio se tratan temas relativos al marco regulador aplicable al proyecto, como: la legislación, estándares técnicos y propiedad intelectual. Respecto a la legislación es importante destacar que la tecnología NFV no cuenta con una regulación específica, ya que es prematuro considerar esta tecnología emergente bajo el marco regulatorio de las comunicaciones electrónicas. Sin embargo, se prevé que el área más probable, donde las nuevas reglas pueden ser necesarias, es la regulación de acceso a las redes virtuales. Como OSM nació como una iniciativa del ETSI, que es una organización de estandarización de la industria de las telecomunicaciones en Europa, el análisis de los estándares técnicos involucrados en el proyecto tiene un papel muy relevante. OSM tiene que seguir los modelos del NFV definidos por el ETSI, tanto los modelos de información como los modelos de datos y también tiene que seguir los estándares de NFV-MANO.

Otro de los aspectos analizados en el estudio es el entorno socio-económico, que es especialmente relevante en los temas relacionados con NFV, debido a las grandes ventajas económicas que se espera obtener al implementar esta tecnología. Esto se debe principalmente a que la inversión realizada en el despliegue de los equipos de las redes tradicionales es alta y se necesitan varios años para amortizarlo, sin embargo, las empresas se ven obligadas a actualizar sus equipos cada pocos años para adecuarse a las nuevas tecnologías o a las demandas de sus usuarios, por lo que les cuesta mucho recuperar la inversión realizada. Por otro lado, con NFV, la inversión que necesita hacer una empresa en su infraestructura de red es única, ya que con implementar varios servidores genéricos se pueden crear todas las VNFs necesarias. Si en algún momento hay que cambiar una función de red o actualizarla, se hace a través de software, que es siempre más económico que comprar nuevo equipamiento.

El estudio finaliza con las conclusiones que se han extraído tras evaluar a OSM, haciendo especial énfasis en las virtudes y defectos de la plataforma que han sido detectados durante su uso. Dentro de las virtudes destacan dos aspectos: el primero, son los pocos recursos hardware que precisa, ya que se ha producido una reducción muy significativa en el consumo de recursos hardware por parte de la plataforma, y el segundo es la simplificación en la instalación de los componentes de la arquitectura, que supone que OSM sea más accesible y pueda ser utilizada por más usuarios y operadores de red; ambos avances han surgido a raíz de la salida de la última versión de la plataforma. Como contrapartida, los defectos que destacan son dos: la pérdida de algunas funcionalidades interesantes respecto a versiones anteriores de la plataforma, como es el caso de una

herramienta que permitía crear las VNFs y NS de una manera más sencilla, y la falta de documentación a la hora de instalar aplicaciones complementarias, aspecto que llama mucho la atención ya que el resto de documentación proporcionada por OSM es muy clara y completa. En cualquier caso, OSM cuenta con muchas más virtudes que defectos y sigue evolucionando, sacando nuevas versiones al mercado cada pocos meses para seguir mejorando.

En los últimos años los operadores han estado realizando pruebas sobre la incorporación de las primeras VNFs en sus redes. En la actualidad, los más avanzados, ya están preparados para el lanzamiento de las primeras de ellas y para ello, OSM, pese a ser una plataforma relativamente joven, se postula como una de las alternativas más importantes en la orquestación de servicios en infraestructuras NFV y no cabe duda de que los operadores la tendrán muy en cuenta cuando estén preparados para el lanzamiento de las primeras funciones virtualizadas.

Palabras clave: OSM (Open Source MANO), NFV (Network Function Virtualization), Orquestación, OpenStack, Redes de comunicaciones.

Abstract

Telecommunications industry is constantly evolving and in recent years has faced the need to use more flexible architectures due to new market demands. The different service providers realised that the old infrastructure need to evolve, because in a few years their requirements are going to exceed the limit of traditional networks, especially in terms of information volume and speed. A new technology emerged because of this limitation, known as Network Function Virtualization (NFV), which is a reference framework related to the architecture of networks, oriented to virtualize different network elements.

Traditional network architecture is static, because hardware and software are strongly integrated. On the opposite, NFV allows that the developed software could be deployed on different platforms, allowing multiple network management points, that are intended to be standard and compatible with solutions from different manufacturers. This allows that complex and specialized hardware devices could be replaced by equivalent systems running on generic servers. The objective is to move from isolated solutions that don´t have the possibility of sharing resources to solutions based on shared platforms, which offer flexibility and independence from manufacturers.

To achieve an NFV solution the first step is to implement its architecture, it is formed by three main components: Network Functions Virtualization Infrastructure (NFVI), Virtualized Network Functions (VNFs) and Management and Orchestration (MANO). Although all the components of the architecture are equally important, this project is focused on the study of the MANO block. This is because Open Source MANO (OSM), the main research objective of this project, is an implementation of this component.

MANO is a separate block of the NFV architecture, which interacts with both NFVI and VNFs. It is responsible for the control of all entities in the architecture, therefore, it must know the usage, status, statistics and the rest of the associated parameters of all the elements involved. In the same way that the NFV architecture has three components, the MANO block has the same number of components, they are: VNF Manager (VNFM), NFV Orchestrator (NFVO) and Virtual Infrastructure Manager (VIM).

Once the most relevant elements of NFV technology have already been introduced, it is time to talk about OSM, the main platform of this project. OSM was created as an initiative of the European Telecommunication Standards Institute (ETSI) to develop a free management and orchestration software stack that follows the standards defined for the NFV technology. It was created by a group of companies, mainly service providers and solution vendors. Its main founding members were Telefónica, BT, RIFT.io, Mirantis, Telekom Austria Group, Intel, Telenor and Canonical. The most appropriate definition to qualify OSM is that it is an open source community, whose goal is to offer a quality MANO software stack for NFV technology production. The purpose is to make OSM compatible with any VNF, accessible to all users and independent of the VIM module. Also, the OSM code is developed according to the procedures established for open source code. The latest information of the orchestrator is available on the OSM

development platform, managed by ETSI, so anyone who could be interested can access to it. OSM is constantly evolving and since it was released at the end of 2016, code programmers have made several modifications on OSM in order to adapt it to the advances in the sector and improve its functionality. Every 6 months, more or less, a new version is released that includes significant changes than the previous one. The most current version is the number four, better known as OSM FOUR, which has been released at the beginning of May of 2018.

OSM has a modular architecture, which has been constantly evolving to adopt the design principles of the cloud. Each one of the modules performs specific and independent functions, which together offer the full functionality of an NFV orchestration platform. Each module of the architecture is installed on a container, that are a set of processes that are separated from the rest of the system. They are like isolated "boxes" that have only the necessary resources to run a specific application or program. Inside the architecture of OSM there are several modules, they are explained in more detail in the thesis. However, there is one of them that deserves to be explained before the others, despite the fact it is not the most important module of the architecture, OSM has given to Monitoring module (MON) a lot of importance after the release of its latest version. It was only an experimental component with very limited functionalities, however, it has acquired a greater importance, becoming part of the main architecture of OSM. There are many monitoring tools on the market and the intention of the MON module is not to compete with them, it pretend to act as an intermediary between these tools and the orchestrator. Therefore, its function is to integrate a specific monitoring tool, chosen according to the needs of the project, with the OSM architecture.

Another element of special relevance is the VIM module, which is an essential element in the MANO architecture, nevertheless, OSM does not implement it. The reason for not including this module is not to impose a specific VIM to work with its architecture, allowing the users to choose what platform is most suitable for each project. There are multiple alternatives, although, the OSM working group only recommends the four that have been tested and verified after being integrated with the orchestration platform, they are: OpenVIM, VMware vCloud Director, Amazon Web Services and OpenStack, this one is the platform chosen to implement the VIM module in the project.

OpenStack is the most important and fastest growing open source cloud computing platform in recent years. It is designed to deliver virtual machines and cloud storage infrastructures. The OpenStack architecture consists of several nodes, that perform different functionalities. The implementation of all of them is not obligatory and depends on the needs of each specific project. There are approximately ten different types of nodes that can be part of the OpenStack architecture, but there are only two of them that are required to use in any implementation: the controller node and the compute node. For the functionality that OpenStack must do in this project, it is only needed to implement these two nodes. On the one hand, the controller node is where the architecture's intelligence resides, it is responsible of coordinating the rest of the nodes and it is where most of the services are installed. On the other hand, the compute node is where the virtual instances

are executed, and it takes care of the minimum services needed to manage those instances. Each service of OpenStack does a very specific task and the sum of several of them is what makes the platform work. Some of these services are essential for their own functioning and there are others that are completely optional, they offer interesting features that can be added depending on the needs of each specific implementation.

The objective of the thesis is to analyse OSM in depth, a purely theoretical study is insufficient, so the best way to study the platform is to analyse it in an experimental way. To a practical study, the OSM architecture must be implemented and the installation procedure of the platform is very important for it. Furthermore, it is also essential to do the OpenStack installation because to correctly evaluate OSM, the platform must have associated a VIM module. Another objective, with a secondary role, but also an important one, is to create a self-contained thesis, this means that everything that is necessary to to replicate the study is documented in the thesis, without the need to look for information in external documents.

Before installing the two platforms needed to perform this thesis, it is important to define how they will be implemented on the equipment that has been used to do the study. Both can be installed directly on a computer or on a virtual machine (VM), being the last option the one that has been chosen for security and portability reasons. Once the virtual machines option is chosen as the most appropriate to the project, it is necessary to use a virtualization tool to create the VM. There are several alternatives in the market, so it is necessary to make an analysis of which is the best option to make this study, prioritizing the performance and economic aspect.

To install OSM there is only one procedure, it consists of using a script that allows to execute hundreds of code sentences automatically. OpenStack is a different case because there are two alternatives to install it: using the DevStack script or doing the installation manually. Although the easiest and fastest way is to get the platform by running the Devstack script, the best option in this case is to do installation manually. The main reason for making this decision is because to implement the VIM module is only necessary some OpenStack services and that can only be done through manual installation, so if Devstack would be used, more services would be created than the necessary ones and it would not make efficient use of hardware resources which are very limited.

Once all components are running on their respective virtual machines, the potential of the OSM platform can be evaluated on the installation, using for this purpose some scenarios known as Network Services (NS). First of all, deploying these NS allows to verify the correct installation and integration of all the components of the implemented architecture and secondly, it allows to evaluate the potential of the orchestrator, that is one of the main objectives of this study.

Three NS are going to be evaluated, the first of the them has been provided by OSM it is very simple and only serves as an example, because that it does not allow doing a correct evaluation of the orchestrator. However, it is excellent as a first contact with the platform and it allows verifying that the installation of the OSM modules has been done

correctly. Once the example NS has been tested and it has been verified that OSM works as it should, another more complex NS can be instantiated, that they allow to evaluate the potential of OSM. These two new NS do not exist and need to be designed and created, that allows a better understanding of how they work and evaluate services with an interesting purpose. In this case, both NS are focused on video re-encoding, that it can be done in many ways, some examples are: modifying the rate at which the video is transmitted, changing the transmission mode… Both NS have two VNFs, one of them transmits the video and the other one modifies the transmission, what it means, it re-encodes the video. The difference between the two NS is in how the video is transmitted, in the simplest case users can only view the video at the same time as it was being broadcast, it is a very similar case to how television works. Meanwhile, in the most complex case, the video can be viewed at any time because it is transmitted like a video on demand server, it is a very similar case to how YouTube works.

Once the potential of the platform has been determined with the NS, an evaluation of OSM can be made, it is focused on an analysis of the platform's performance. For this purpose, the deployment times of the OSM platform, a simple NS and complex NS are evaluated. Rare features that OSM incorporates are also tested, and they differentiate it from other orchestrators. These functionalities are the inclusion of complementary applications in the platform installation and the VIM-emulator functionality.

One of these complementary applications is Grafana, it allows visualize graphically some values about the instances made in OSM, such as CPU utilization, received packets, etc. The integration with Grafana is done to promote the use of the MON module, it has acquired a great protagonism following the launch of the new version of OSM, as it was mentioned before. The reason is because MON simply provides the data in numerical form, difficult for people to use, so integration with Grafana, or other similar applications, helps to interpret the data obtained easily. Another of these applications is Kibana, it allows graphically display some OSM data, but in this case, it is a way of displaying events. Kibana provides the freedom to select how data is displayed, users can choose: histograms, bar graphs, pie charts, and some more sophisticated graphs. Its use provides a better understanding of OSM statistics and even the detection of anomalous behaviour in the platform. Apart from that, the VIM-emulator functionality makes possible to do not having to associate a VIM with the OSM architecture, because this module is emulated on the platform itself. This is a very useful function because it allows to get quickly a complete MANO architecture, without having to create a VIM, which is a complex process.

The study also explains topics related to the regulatory framework applicable to the project, such as: legislation, technical standards and intellectual property. About legislation, it is important to note that the NFV technology does not have a specific regulation, due to the fact that it is considered premature to evaluate this emerging technology under the review of the regulatory framework for electronic communications. However, it is anticipated that the most likely area where new rules may be needed is the regulation of access to virtual networks. About technical standards, as OSM was born as

an initiative of ETSI, which is an organization for the standardization of the telecommunications industry in Europe, the analysis of the technical standards involved in the project has a very important role. Open Source MANO must follow the NFV models defined by ETSI, information models and data models, and it has to follow the NFV-MANO standards.

Another of the aspects analysed in the thesis is the socio-economic environment, it is especially relevant in technologies related with NFV, due to the great economic advantages that are expected to be obtained by implementing it. This is mainly due by fact that the investment made in the deployment of traditional network equipment is high and it takes several years to amortize it, however, companies are forced to update their equipment every few years to adapt to new technologies or the demands of their users, so it is very difficult for them to recover the investment made. A totally different scenario occurs with NFV technology because the investment that a company needs to make in its network infrastructure is unique, they only need to use several generic servers that can create all the necessary VNFs. Any time they need changing a network function or being updated, it is done by software, that it always will be cheaper than buying new equipment.

The study is completed with the conclusions that have been learned after evaluating OSM, with special emphasis on the strengths and weaknesses of the platform that have been detected during its use. On the one hand, the following two virtues stand out: the first of them is the few hardware resources that it needs, due to a very significant reduction in the consumption of hardware resources by the platform, and the second of them is the simplification in the installation of the components of the architecture, this makes OSM more accessible and it could be used by more users and network operators; both advances have been made by the latest version of the platform. On the other hand, the two defects that stand out are: the first of them is the loss of some interesting functionalities compared to previous versions of the platform, such as the case of a tool that made possible to create VNFs and NS in a simpler way, and the second of them is the lack of documentation when installing complementary applications, an aspect that stands out because the rest of the documentation provided by OSM is very clear and complete. Anyway, OSM has many more virtues than defects and it continues evolving, bringing new versions to market every few months to continue improving.

In the past few years, operators have been testing the incorporation of the first VNFs into their networks. Nowadays, the most advanced ones are already prepared for the launch of the first VNFs and for this reason, OSM, despite being a relatively young platform with only two years of life, is postulated as one of the most important alternatives in the orchestration of services in NFV infrastructures and there is no doubt that operators will take it into account when will be ready to launch the first virtualized functions.

Key words: OSM (Open Source MANO), NFV (Network Function Virtualization), Orchestration, OpenStack, Communication networks.

Índice de contenido

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se aborda la motivación que ha llevado a plantear y desarrollar este Trabajo Fin de Grado sobre el estudio de la plataforma de orquestación NFV Open Source MANO, así como los objetivos del mismo, el marco regulador y la estructura de este documento.

1.1. Motivación del proyecto

La industria de las telecomunicaciones está en constante evolución y en los últimos años se ha encontrado ante la obligación de utilizar arquitecturas más flexibles debido a las nuevas exigencias del mercado. Los diferentes proveedores de servicios se dieron cuenta de que era necesario desarrollar las infraestructuras, puesto que los requisitos requeridos a corto plazo iban a exceder el límite de las redes tradicionales, sobre todo en volumen de información y velocidad. Debido a dicha limitación surge Network Function Virtualization (NFV) [1] un marco de referencia relacionado con la arquitectura de las redes, orientado a virtualizar diferentes elementos dentro de las mismas.

La arquitectura de las redes tradicionales es estática, debido a que el *hardware*[1] (HW) y el *software*[2] (SW) están estrechamente integrados. Por el contrario, NFV permite que el software desarrollado se despliegue en plataformas diferentes, permitiendo varios puntos de gestión de la red, compatibles con soluciones de varios fabricantes. De esta manera se consigue sustituir dispositivos hardware complejos y especializados por sistemas equivalentes que se ejecutan sobre servidores genéricos. Se pretende pasar de soluciones aisladas sin posibilidad de compartir recursos, a soluciones basadas en plataformas compartidas, que ofrecen libertad a la hora de seleccionar los fabricantes y flexibilidad.

La motivación de este proyecto es comprender este nuevo marco de referencia que pretende dejar obsoletas las redes de comunicaciones tal y como son actualmente. El estudio se centrará en el bloque MANO, el cual es vital dentro de la arquitectura NFV, ya que en él reside toda la inteligencia, encargándose de la gestión y orquestación del resto de bloques para que todo funcione correctamente.

1.2. Objetivo del proyecto

El objetivo es la realización de un estudio exhaustivo de la plataforma de orquestación NFV Open Source MANO (OSM). Por lo tanto, este Trabajo Fin de Grado (TFG) no se limita a un simple estudio teórico, sino que gran parte del trabajo es a nivel práctico, ya que la mejor manera de estudiar la plataforma es instalarla. Una vez que esté todo funcionando se puede evaluar sobre la implementación el potencial de OSM, usando para ello, una serie de escenarios especialmente diseñados para determinar el alcance de su funcionalidad.

Los hitos del proyecto son:

- Implementación de OSM.
    - Creación de una máquina virtual (VM) con el sistema operativo (OS) Ubuntu 16.04.
    - Instalación de OSM sobre la máquina virtual.
- Implementación de OpenStack.

[1] Elementos físicos que componen un sistema informático.
[2] Elementos lógicos que componen un sistema informático, como los programas, los datos …

    - Creación de dos máquinas virtuales con el sistema operativo Ubuntu 16.04.
    - Instalación del nodo controller en una máquina virtual.
    - Instalación del nodo compute en la otra máquina virtual.
- Integración de OSM y OpenStack.
- Realización de pruebas sobre la arquitectura completa.
    - Instanciación de un escenario básico proporcionado por OSM.
    - Diseño de escenarios complejos.
    - Instanciación de los escenarios complejos.
    - Evaluar funcionalidades que diferencian a OSM.

Un objetivo secundario, pero también importante, es crear un TFG autocontenido, es decir, que en la memoria quede documentado todo lo necesario para poder replicar el estudio sin necesidad de buscar información en documentos ajenos.

1.3. Marco regulador

Este apartado contiene un análisis del marco regulador aplicable al proyecto que incluye: un análisis de legislación, un análisis de los estándares técnicos y un análisis de la propiedad intelectual.

1.3.1. Análisis de la legislación

La Comisión Europea ha publicado el informe [2] sobre las implicaciones tecnológicas, económicas y regulatorias de las tecnologías emergentes Software Defined Networks (SDN) y NFV, que se cree que jugarán un papel importante en el futuro panorama de las telecomunicaciones. En dichos informes, se explica que es prematuro considerar estas tecnologías bajo el marco regulatorio de las comunicaciones electrónicas. Sin embargo, los expertos involucrados en dichas investigaciones destacaron que el área más probable, donde las nuevas reglas pueden ser necesarias, es la regulación de acceso a las redes virtuales.

Por lo tanto, al no contar NFV con una regulación específica, el bloque MANO tampoco cuenta aún con ella y mucho menos la plataforma OSM.

Otro caso muy distinto es OpenStack, como plataforma de *Cloud Computing*[3] debe cumplir los requisitos establecidos por la Ley 34/2002 de Servicios de la Sociedad de la Información y del Comercio Electrónico (LSSI) [3]. Algunos de esos requisitos son:

- Garantizar la seguridad de la información del cliente, tomando las medidas técnicas y organizativas necesarias que eviten su alteración, pérdida o acceso no autorizado.
- El uso de los datos de carácter personal requiere el consentimiento explícito por parte del cliente.

[3] Cloud Computing: ofrecer servicios a través de internet.

- Los datos personales deben ser eliminados cuando dejen de ser necesarios para el fin para el que hubiesen sido registrados.

1.3.2. Estándares técnicos

El European Telecommunications Standards Institute (ETSI) ha creado un Industry Specification Group (ISG) para la tecnología NFV [4] con el fin de lograr la arquitectura común necesaria para soportar las Virtualized Network Functions (VNFs) a través de un enfoque coherente.

NFV-ISG cuenta actualmente con cuatro grupos de trabajo:

- Arquitectura de Infraestructura.
- Gestión y Orquestación.
- Arquitectura de Software.
- Fiabilidad y Disponibilidad.

OSM pertenece al grupo de "Gestión y Orquestación", también conocido como MANO, que tiene como objetivo crear una pila de código abierto que siga los modelos NFV del ETSI, tanto los modelos de información como los modelos de datos. Eso implica:

- Ser capaz de seguir los modelos publicados para el servicio y despliegue de NFV, además de ampliarlos y recomendarlos de vuelta a ETSI NFV.
- Asegurar un comportamiento predecible de VNF y Network Service (NS).
- Habilitar un ecosistema de soluciones VNF basadas en los modelos del ETSI-NFV que esté listo para ser ofrecido a los proveedores de Cloud Computing y de servicios.

A su vez, la plataforma de orquestación OSM tiene que seguir los estándares de NFV-MANO, lo cuales definen:

- Los bloques de la arquitectura y como se deben relacionar, explicado en el apartado 2.1.1.3.
- Las entidades NFV que hay que implementar y gestionar, explicadas en el apartado 2.1.2.
- Los descriptores que hay que usar, explicados en el apartado 2.3.2.

1.3.3. Propiedad intelectual

El Estatuto del Estudiante, aprobado en el año 2010 por el Ministerio de Educación, recoge uno de los derechos que tienen los estudiantes en cuanto al tema de la propiedad intelectual [5]: "Al reconocimiento de la autoría, de los trabajados elaborados durante sus estudios y a la protección intelectual de los mismos".

Por otro lado, la Ley de Propiedad Intelectual (LPI) [6] en su artículo 5 dice que se considera autor a la persona natural que crea alguna obra literaria, artística o científica, y por lo tanto es a quien corresponden los derechos de propiedad intelectual de la misma.

Existen una serie de casos especiales a la hora de realizar un TFG, en los que la autoría del mismo, y por lo tanto los derechos de propiedad intelectual, no están tan definidos. Estos casos especiales son:

- Convenio con una empresa privada.
- Trabajo realizado entre varias personas.
- Trabajo que incluye alguna patente.

Como este trabajo no pertenece a ninguno de esos casos, la propiedad intelectual del mismo corresponde al autor.

1.4. Estructura de la memoria

En este apartado se detalla la estructura del documento, el cual está dividido en siete capítulos. Se incluye un resumen de cada uno para facilitar su posterior lectura.

- **Capítulo 1 – Introducción:** En este capítulo se expone la motivación del proyecto, los objetivos del mismo, el marco regulador y la estructura de este documento.
- **Capítulo 2 – Estado del arte:** Capítulo en el que se describen los fundamentos teóricos de las tecnologías usadas para el desarrollo del proyecto.
- **Capítulo 3 - Montaje de la arquitectura:** En este capítulo se describe el procedimiento de instalación de los componentes que forman parte de la arquitectura necesaria para desarrollar el proyecto.
- **Capítulo 4 - Creación de Network Services y evaluación de resultados:** En este capítulo se diseñan diferentes NS que permiten determinar el potencial de OSM. También se realiza una evaluación de la plataforma, la cual se centra en un análisis del rendimiento y la prueba de funcionalidades poco comunes que lo diferencian del resto de orquestadores.
- **Capítulo 5 – Planificación y análisis del entorno socio-económico:** En este capítulo se muestra la planificación realizada para poder llevar a cabo el proyecto y un análisis del entorno socio-económico.
- **Capítulo 6 – Conclusión y trabajos futuros:** En este capítulo se exponen las conclusiones obtenidas de la realización del proyecto y se proponen nuevas líneas de investigación para trabajos futuros sobre OSM.
- **Capítulo 7 – Anexos:** En este capítulo se recogen todos los procedimientos que, aunque importantes para llegar a la solución, no tenían cabida en otros capítulos.

Capítulo 2

Estado del arte

En este capítulo se describen los fundamentos teóricos del proyecto. Incluye qué es la tecnología NFV y se explican los componentes que forman su arquitectura, dando especial relevancia al bloque MANO, debido que OSM es una implementación de dicho bloque. Se incluye el estudio teórico de OSM, en el cual se explica qué es la plataforma, cómo funciona y los módulos que forman su arquitectura, entre los cuales destaca el módulo VIM, siendo OpenStack la plataforma elegida para implementarlo, la cual también es analizada en el estudio.

2.1. Network Function Virtualization

Network Function Virtualization es un término que surgió a raíz de que se presentara el whitepaper homónimo [1] por el ETSI en el año 2012. Es un modelo de arquitectura de red que tiene como objetivo la virtualización de servicios de la misma. Permite la sustitución de dispositivos de hardware especializados y caros, como routers, *switches*[4] o *firewalls*[5], por funciones de red basadas en software que se ejecutan como máquinas virtuales en servidores genéricos. Esto implica que todas las funciones realizadas por los nodos de la red deben estar definidas en un modelo virtual, más conocidas como VNFs.

Esta nueva tecnología supone una gran ventaja respecto a las redes tradicionales, ya que se proporcionan los servicios en el momento que se necesitan y en el lugar donde se requieren, por lo que se hace un uso eficiente de los recursos. Si un cliente necesita una nueva VNF, el operador puede simplemente activar una nueva VM para realizar esa función y con la misma facilidad con la que se activó la máquina virtual, se puede desactivar cuando ya no sea necesaria. Por ejemplo, en lugar de desplegar un nuevo equipo hardware para proporcionar seguridad, un proveedor puede implementar el software diseñado para proporcionar dicha seguridad en un servidor que ya estuviera instalado en la red. Es importante destacar que una VNF en sí misma no es una solución NFV, sino que es la agrupación de varias VNFs las que permite obtener el segmento de red virtualizado.

2.1.1. Arquitectura NFV

En una arquitectura NFV existen tres componentes, como se demuestra en [7]: Network Functions Virtualization Infraestructure (NFVI), Virtualized Network Functions y MANO. En la Fig. 2.1 se muestra dicha arquitectura.

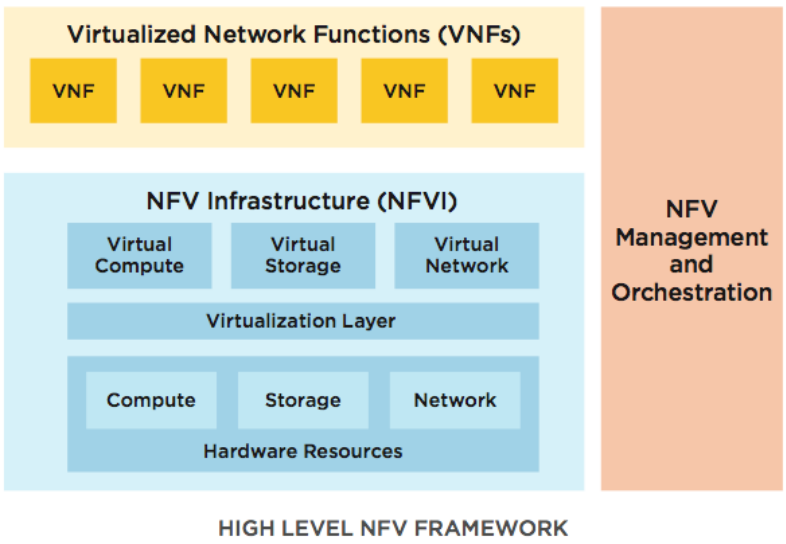

Fig. 2.1. Arquitectura NFV [8]

[4] Dispositivo de interconexión utilizado para conectar equipos en red.
[5] Dispositivo de seguridad de la red que decide si debe permitir o bloquear un tráfico específico.

2.1.1.1. Network Functions Virtualization Infraestructure (NFVI)

El bloque NFVI describe los componentes de hardware y software sobre los que se construyen las redes virtuales. Contiene el HW para alojar las máquinas virtuales, el SW que hace posible la virtualización y los propios recursos virtualizados.

2.1.1.2. Virtualized Network Functions (VNFs)

Las VNFs utilizan las máquinas virtuales que ofrece el bloque NFVI, añadiendo el software necesario sobre ellas para obtener las funciones virtualizadas de red.

2.1.1.3. Management and Orchestration (MANO)

Es importante aclarar que todos los bloques de la arquitectura NFV son igual de importantes, sin embargo, MANO es el que se aborda con más profundidad en el TFG. Esto se debe a la gran relevancia que tiene para este proyecto, ya que OSM es una implementación de este bloque.

MANO es un bloque separado dentro de la arquitectura NFV, que interactúa tanto con la NFVI como con las VNFs. Es el responsable del control de todas las entidades dentro de la arquitectura NFV. Por tanto, debe conocer el uso, el estado, las estadísticas y el resto de los parámetros asociados de todos los elementos implicados en la arquitectura.

A continuación, se describen los bloques funcionales de una arquitectura MANO [9], los cuales están representados en la Fig. 2.2.

- **VNF Manager (VNFM):** Se encarga de la gestión de las VNFs. Sus funciones más relevantes son:
  - Gestionar el ciclo de vida de las VNFs, es decir, se encarga de crear, mantener y terminar las instancias.
  - Gestionar los fallos, la configuración, el rendimiento y la seguridad de las VNFs.
  - Reducir o aumentar el uso de la CPU de las VNFs, dependiendo de las necesidades del sistema.
- **NFV Orchestrator (NFVO):** Este componente se encarga de gestionar los NS. Por lo tanto, se encarga de:
  - Gestionar el ciclo de vida de los NS.
  - Crear un servicio de extremo a extremo entre diferentes VNFs.
  - Crear instancias de VNFM, donde corresponda.
  - Validar y autorizar las solicitudes del bloque NFVI.
- **Virtual Infrastructure Manager (VIM):** Este módulo gestiona los recursos de NFVI en un dominio. Puede haber varios VIM en una arquitectura NFV y cada uno de ellos se encarga de gestionar su respectivo dominio NFVI, a este fenómeno se le conoce como multi-VIM. Las funciones más relevantes de este bloque son:

- Gestionar el ciclo de vida de los recursos virtuales en un dominio NFVI. Por lo tanto, se encarga de la creación, mantenimiento y destrucción de las máquinas virtuales en los recursos físicos de un dominio NFVI.
- Mantener un inventario de las VMs asociadas a los recursos físicos de los que están haciendo uso.
- Gestionar los errores asociados al hardware, al software y a los recursos virtuales.

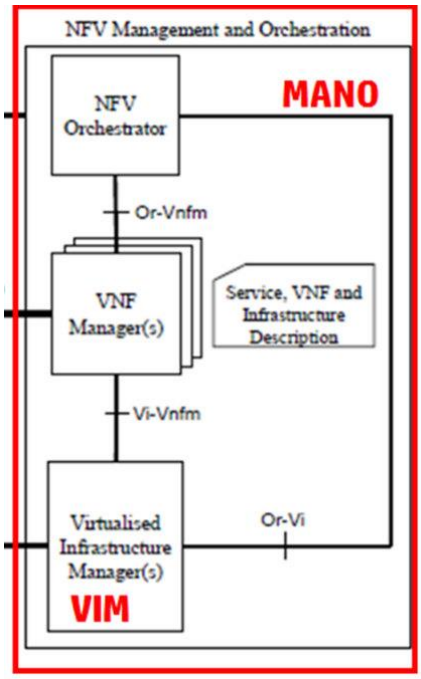

Fig. 2.2. Arquitectura MANO [10]

2.1.2. Entidades NFV

Dentro de una solución NFV existen dos entidades [9] que deben estar siempre presentes y que deben gestionarse e implementarse:

- **Virtual Network Function (VNF):** Es un software que realiza una función de red concreta en entornos cloud, como por ejemplo un router o un switch. La combinación de varias VNFs sirve para implementar un segmento de red virtualizado. Una VNF está compuesta siempre por los tres siguientes elementos (al menos uno de cada tipo), los cuales aparecen representados en la Fig. 2.3:
  - **Virtual Deployment Unit (VDU):** Es la máquina virtual que aloja a la función de red. Especifica la imagen, la capacidad de almacenamiento, la Random Access Memory (RAM) y el número de CPUs de la VM.
  - **Virtual Link (VL):** Proporciona conectividad entre VDUs.
  - **Connection point (CP):** Se utiliza para conectar los VLs con los VDUs.

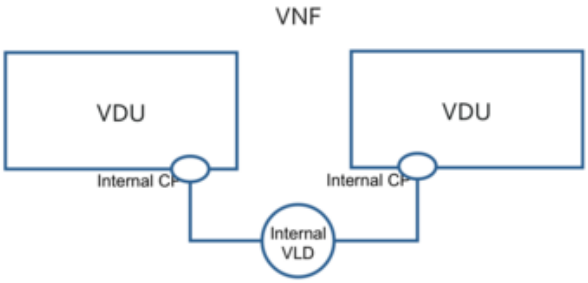

Fig. 2.3. Elementos de una VNF [11]

- **Network Service (NS):** Es una composición de VNFs. El comportamiento de un NS es el resultado de la combinación de VNFs individuales y la NFVI. Está compuesta siempre por al menos una VNF, comportamiento que queda reflejado en la Fig. 2.4.

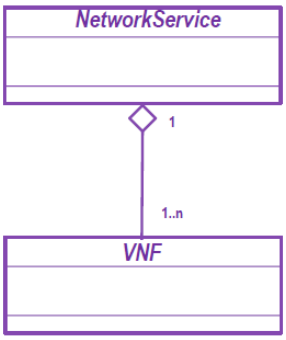

Fig. 2.4. Diagrama de dependencias de las entidades NFV

2.2. Open Source MANO (OSM)

OSM nació como una iniciativa del ETSI, como se demuestra en [12], para desarrollar una pila de software libre de gestión y orquestación que siga los estándares definidos por ETSI para la tecnología NFV. En la Fig. 2.5 se muestra como la plataforma interactúa con los demás componentes de una arquitectura NFV.

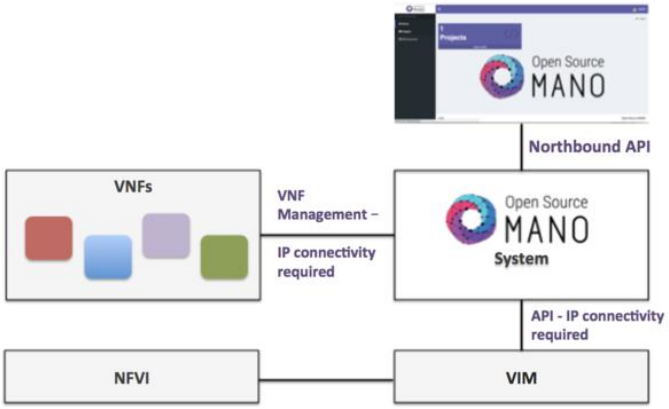

Fig. 2.5. Arquitectura genérica de OSM [13]

La definición más acertada para definir a OSM es la de una comunidad de código abierto, cuyo objetivo es ofrecer una pila de software MANO de calidad para producción de tecnología NFV. Pretende ser compatible con cualquier VNF, accesible para todos los usuarios e independiente del módulo VIM.

Fue creado por un conjunto de empresas, principalmente proveedores de servicio y vendedores de soluciones. Entre sus miembros fundadores se encuentran Telefónica, BT, RIFT.io, Mirantis, Telekom Austria Group, Intel, Telenor y Canonical.

OSM está en constante evolución y desde su nacimiento, a finales de 2016, ha sufrido varias modificaciones con el fin de adaptarse a los avances del sector y mejorar su funcionalidad. Cada 6 meses aproximadamente sale al mercado una nueva versión que incluye cambios notables respecto a la anterior. La versión más reciente es la número cuatro, más conocida como OSM FOUR, que salió a principios de mayo de 2018.

El código de OSM está desarrollado de acuerdo con los procedimientos establecidos para el código abierto. En la plataforma de desarrollo de OSM, gestionada por la ETSI, se encuentra la información más reciente, de esta manera cualquier usuario interesado pueda acceder a ella fácilmente.

En los últimos años los operadores han estado realizando pruebas sobre la incorporación de las primeras VNFs en sus redes. En la actualidad, los más avanzados, ya están preparados para el lanzamiento de las primeras de ellas y para ello, OSM se presenta como una de las mejores alternativas del mercado para la orquestación de servicios NFV.

2.2.1. Arquitectura de OSM

OSM tiene una arquitectura modular [14] que ha ido evolucionando constantemente para adoptar los principios de diseño originales de la nube. Cada uno de los módulos realiza funciones concretas e independientes, que puestas en conjunto ofrecen la funcionalidad completa de una plataforma de orquestación NFV.

Cada módulo de la arquitectura está instalado sobre un contenedor, el cual es un conjunto de procesos que están separados del resto del sistema. Los contenedores son como “cajas” aisladas que disponen únicamente de los recursos imprescindibles para poder ejecutar una determinada aplicación o programa. En la Fig. 2.6 se muestra un esquema de cómo se ejecutan los contenedores. OSM utiliza dos tecnologías distintas para implementarlos:

- **Dockers:** Es la tecnología para representar los contenedores a partir de la versión FOUR. Todos los módulos de la arquitectura son compatibles con esta tecnología.
- **Linux Containers (LXC):** Es la forma tradicional de OSM para representar los módulos de su arquitectura y, aunque se ha migrado a los contenedores Dockers, algunos componentes siguen haciendo uso de esta tecnología. LXD es el controlador de los contendores LXC.

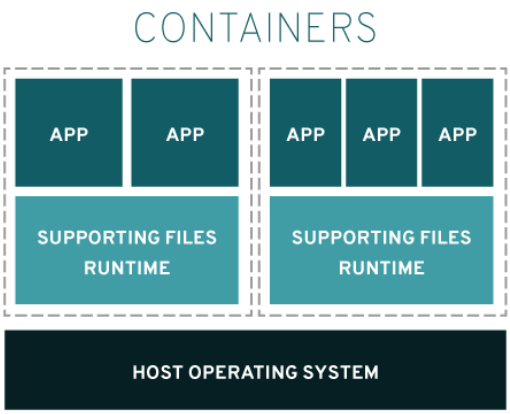

Fig. 2.6. Contenedores [15]

Durante la versión FOUR, la comunidad de OSM adaptó los módulos para conseguir un rápido progreso en la configuración de VNFs y también ha continuado evolucionando tanto la arquitectura interna como la externa para satisfacer las necesidades de los operadores de red.

A continuación, se presentan las características más relevantes de los componentes que forman parte de la arquitectura de OSM, que están representados en la Fig. 2.7. Sin embargo, hay que tener en cuenta que está arquitectura solo es aplicable a la última versión, la FOUR, ya que tanto el número de módulos, como las funciones que desempeñan han variado a lo largo de las diferentes versiones.

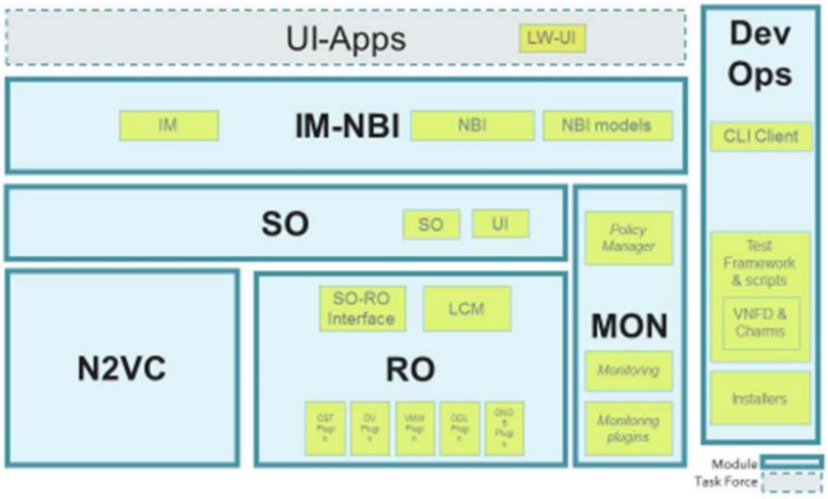

Fig. 2.7. Módulos de la arquitectura de OSM [14]

2.2.1.1. User Interface (UI)

Este módulo permite a los usuarios gestionar el resto de los servicios proporcionados por OSM. El componente principal del módulo es Launchpad, una interfaz gráfica interactiva. Sus funciones más relevantes son:

- Mostrar estadísticas en tiempo real de las VNFs y NS.
- Mostrar una vista detallada de las topologías de cómputo y red.
- Administrar las credenciales de acceso para los entornos VIM.

2.2.1.2. Service Orchestrator (SO)

Es el componente de orquestación principal en el sistema, dirige el flujo de trabajo a través de OSM y es, por lo tanto, responsable de:

- Gestionar el ciclo de vida y la ejecución primitiva del servicio.
- Proporcionar soporte a los proyectos y a los usuarios, y de la implementación de controles de acceso basados en roles.
- Validar que los descriptores Network Service Descriptor (NSD) y Virtual Network Function Descriptor (VNFD) se ajustan al esquema definido.
- Soportar las operaciones del ciclo de vida de los NS y VNF, que son actualizar, crear, leer y eliminar.

2.2.1.3. Resource Orchestrator (RO)

Módulo encargado de la gestión de los recursos. Sus funciones más importantes son:

- Gestionar y coordinar las asignaciones de recursos.
- Localizar los recursos que pueden estar en múltiples VIMs y en múltiples controladores SDN.

2.2.1.4. VNF Configuration and Abstraction (VCA)

Este módulo es responsable de:

- Habilitar las configuraciones, acciones y notificaciones relacionadas con las interacciones entre VNFs.
- Realizar la configuración inicial de las VNFs, por lo tanto, este módulo se puede considerar como un VNFM genérico con un conjunto limitado de características.

2.2.1.5. Network Service to VNF Communication (N2VC)

El módulo N2VC es responsable únicamente de la comunicación entre los módulos SO y VCA. Pese a encargarse únicamente de esa función, es un bloque vital dentro de la arquitectura.

2.2.1.6. Kafka bus

Una de las nuevas aportaciones más relevantes introducidas en la versión FOUR es la inclusión bus de mensajería basado en Kafka, el cual proporciona un nuevo canal dedicado para la comunicación asíncrona entre los componentes de la arquitectura, como se muestra en la Fig. 2.8. Esto hace que OSM se pueda integrar más fácilmente con nuevos módulos y ha facilitado una mayor centralización de los servicios comunes.

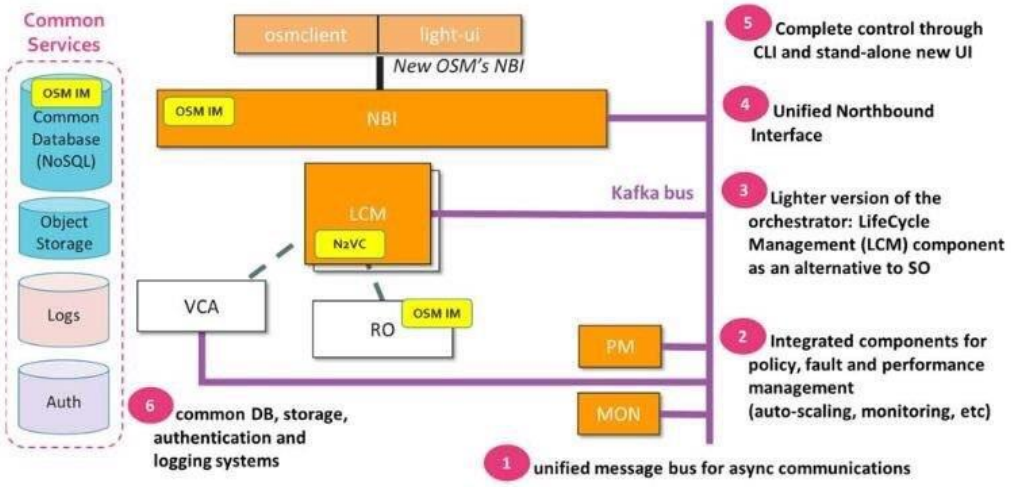

Fig. 2.8. Kafka bus [14]

2.2.1.7. Virtual Infrastructure Manager (VIM)

Es un elemento imprescindible dentro de una arquitectura MANO, sin embargo, OSM no lo implementa. La razón de no incluirlo es no imponer un VIM concreto a la hora de trabajar junto a su arquitectura, permitiendo elegir la plataforma que se adecue más a cada proyecto. Existen múltiples alternativas, sin embargo, el grupo de trabajo de OSM únicamente recomienda las cuatro que han sido probadas y verificadas tras haber sido integradas junto a la plataforma de orquestación.

- **OpenVIM** [16]**:** Es una implementación básica de un VIM NFV, que era originalmente parte de OpenMANO, un proyecto anterior a OSM del ETSI para conseguir una implementación MANO.
- **VMware vCloud Director** [17]**:** Es un software de automatización, despliegue y gestión de recursos de infraestructura virtual de entornos en la nube.
- **Amazon Web Services** [18]**:** Es una plataforma de servicios en la nube que ofrece, entre otras funcionalidades, almacenamiento de bases de datos, servicios de computación y entrega de contenido.
- **OpenStack** [19]**:** Es la plataforma Cloud Computing de software libre más importante y que más ha crecido en los últimos años. Está diseñada para ofrecer máquinas virtuales e infraestructuras de almacenamiento en la nube. Es la opción elegida para llevar a cabo este proyecto, por lo que se explica más detenidamente en el apartado 2.4.

2.2.1.8. Monitoring (MON)

Este módulo es uno de los que más ha evolucionado en la última versión. Antes era únicamente un componente experimental con funcionalidades muy limitadas, sin embargo, con el lanzamiento de la versión FOUR ha adquirido un mayor protagonismo, pasando a formar parte de la arquitectura principal de OSM.

Existen muchas herramientas de monitorización en el mercado y la intención del módulo MON no es competir con ellas, sino actuar como intermediario entre dichas herramientas y el orquestador. Por lo tanto, su función es integrar una herramienta de monitorización concreta, elegida según las necesidades del proyecto, con la arquitectura de OSM.

2.2.1.9. Northbound Interface (NBI)

Permite que los componentes de la arquitectura de OSM se comuniquen con el componente de nivel superior, Launchpad en este caso. Ha sido totalmente renovada respecto a versiones anteriores para que esté alineada con la especificación SOL005 de ETSI NFV.

2.2.1.10. Policy Manager (PM)

Es un nuevo componente en la arquitectura de OSM que ha surgido a raíz de la gran importancia que ha adquirido MON, módulo con el cual está muy relacionado. La función principal de PM es proporcionar soporte para añadir notificaciones, las cuales aparecen cuando una nueva métrica o alarma es relevante para el módulo MON.

2.2.2. Descriptores

Para desplegar en el sistema las entidades NFV, descritas en el apartado 2.1.2, OSM utiliza los siguientes descriptores [9]:

- **Network Service Descriptor (NSD):** Es un documento de configuración, que sigue la especificación del ETSI MANO, y determina cómo se componen los NS. Sirve para incorporar e instanciar NS a los sistemas, siendo en este caso OSM.
- **Virtual Network Virtualization Descriptor (VNFD):** Es un fichero que describe una VNF en términos de despliegue y requisitos de comportamiento operativo. También contiene requisitos de conectividad, interfaz y recursos virtualizados. Su función es incorporar VNFs a OSM.

2.2.3. Complementos externos

Otra de las principales mejoras de OSM versión FOUR es la integración con aplicaciones externas. A pesar de que en versiones más antiguas también era posible dicha integración, eran los propios usuarios los que tenían que ingeniárselas para hacerlo posible. El avance de esta nueva versión respecto a las anteriores ha sido ofrecer la posibilidad de incorporar algunas aplicaciones junto con la instalación estándar de OSM, además de aportar la documentación necesaria para integrar dichas aplicaciones.

Existen infinidad de aplicaciones que pueden complementar al orquestador y es imposible que el grupo de trabajo de OSM proporcione la documentación necesaria para integrarlas todas, por lo que se han incluido las que, a su parecer, tienen más sentido usarse junto a la plataforma.

2.2.3.1. Grafana

Una de estas aplicaciones es Grafana [20], la cual permite visualizar gráficamente algunos valores relacionadas con las VNFs desplegadas en OSM, como pueden ser la utilización de CPU, los paquetes recibidos, etc.

La integración con Grafana pretende potenciar el uso del módulo MON, que como ya se comentó anteriormente, ha adquirido un gran protagonismo a raíz del lanzamiento de la nueva versión de OSM. Esto se debe a que MON simplemente proporciona los datos de forma numérica, difíciles de utilizar para las personas. Por lo que la integración con Grafana, u otra aplicación similar, ayuda a interpretar de una manera más sencilla los datos obtenidos.

El proceso de obtención de las gráficas, representado en la Fig. 2.9, es el siguiente:

1. El usuario activa la monitorización, orden que llega al sistema a través del NBI.
2. El módulo MON recibe la orden e incorpora las métricas al Kafka bus.
3. Prometheus, que es un complemento de Grafana, se encarga de recogerlas del bus y almacenarlas.
4. Prometheus proporciona las métricas a Grafana.
5. Grafana convierte los datos numéricos en gráficas.

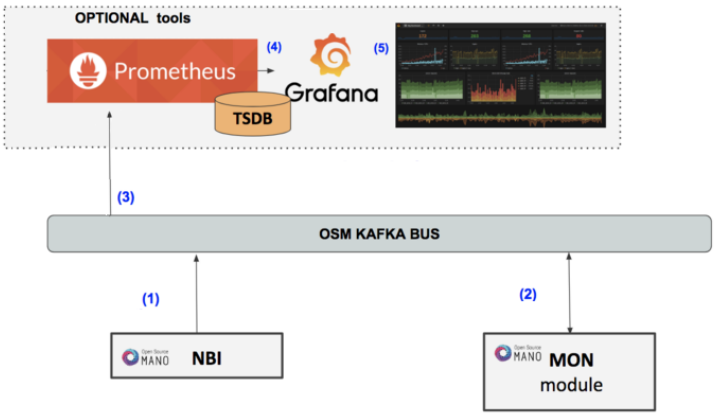

Fig. 2.9. Procedimiento de obtención de gráficas en Grafana [21]

2.2.3.2. Kibana

Kibana [22], al igual que Grafana, permite visualizar gráficamente algunos datos de OSM, pero en este caso no se trata de representar valores de utilización, sino de la visualización de eventos. Kibana proporciona la libertad de seleccionar la forma en que se visualizan los datos, pudiendo elegir: histogramas, gráficos de barras, gráficos circulares y algunas gráficas más sofisticadas. Como indica la propia plataforma [22]: "Una imagen vale más que mil líneas de registro", por lo que su uso permite comprender mejor las estadísticas de OSM e incluso detectar comportamientos anómalos dentro de la plataforma.

2.3. OpenStack

OpenStack, cuyo logo aparece en la Fig. 2.10, es la plataforma Cloud Computing de software libre más importante y que mayor crecimiento ha experimentado en los últimos años. Está diseñada para ofrecer máquinas virtuales e infraestructuras de almacenamiento en la nube.

Fig. 2.10. Logo de OpenStack [23]

El papel de OpenStack en este proyecto es hacer del módulo VIM, imprescindible en una arquitectura MANO.

2.3.1. Arquitectura OpenStack

La arquitectura de OpenStack [24] está formada por varios nodos, los cuales llevan a cabo distintas funcionalidades. La implementación de todos ellos no es obligatoria y depende de las necesidades de cada proyecto.

Existen aproximadamente diez tipos de nodos distintos que pueden formar parte de la arquitectura de OpenStack, sin embargo, solo hay dos de ellos que deben estar siempre presentes en cualquier implementación: el nodo controller y el nodo compute. Para las funciones que debe llevar a cabo OpenStack en este proyecto con instalar estos dos nodos es suficiente.

2.3.1.1. Nodo Controller

El nodo controller es dónde reside la inteligencia de la arquitectura, ya que se encarga de coordinar el resto de los nodos y en él se instalan la mayoría de los servicios de OpenStack, normalmente los no computacionales. Ejecuta los siguientes servicios:

- **Servicio de identidad.**
- **Servicio de imagen.**
- **Servicio de computo.** Solo la parte de gestión.
- **Servicio de red.** Principalmente la parte de gestión.
- **Interfaz gráfica.**
- **Algunos servicios de soporte:** Base de datos *SQL*[6], cola de mensajes, *NTP*[7], etc.

[6] Acrónimo de Structured Query Language. Es un lenguaje de programación diseñado para administrar bases de datos.

[7] Acrónimo de Network Time Protocol. Es un protocolo de internet que sirve para sincronizar los relojes de los sistemas informáticos.

2.3.1.2. Nodo Compute

El nodo compute es dónde se ejecutan las instancias virtuales y se encarga de los servicios mínimos necesarios para administrar dichas instancias, que son:

- **Servicio de computo.**
- **Servicio de red.** Ejecuta un agente de servicio de red que conecta las instancias a redes virtuales.

2.3.2. Servicios de OpenStack

Cada servicio en OpenStack se encarga de una tarea muy concreta y la puesta en común de varios de ellos es lo que permite que funcione. Algunos de estos servicios son imprescindibles para el correcto funcionamiento de la plataforma, sin embargo, hay otros que son completamente opcionales, ofreciendo interesantes funcionalidades que pueden ser añadidas dependiendo de las necesidades de cada implementación concreta.

A continuación, se describen los servicios que deben estar siempre presentes en cualquier implementación, los cuales aparecen representados en la *Fig. 2.11*[8]. Estos servicios van a ser los únicos que se han instalado en este proyecto, ya que, sin necesidad de usar servicios opcionales se puede conseguir que OpenStack funcione como el módulo VIM de una arquitectura MANO.

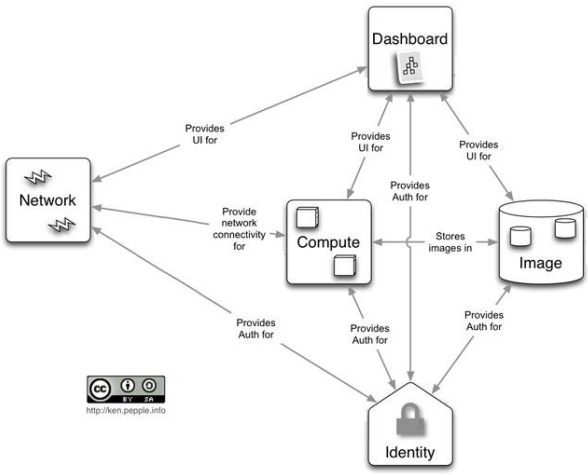

Fig. 2.11. Servicios básicos de OpenStack [25]

2.3.2.1. Servicio de identidad (Keystone)

Keystone [26] es el servicio encargado de gestionar la autenticación de los usuarios, de los tenant y de los demás servicios de Openstack. También se encarga de la gestión de los proyectos, almacenando que usuarios y roles pertenecen a cada uno.

[8] Imagen modificada para ajustarse a los servicios de OpenStack implementados en el proyecto. La imagen está sujeta a la licencia CC que permite dicha modificación.

Para entender el funcionamiento de Keystone es conveniente conocer algunos conceptos:

- **Usuario:** Es la representación de una persona que usa OpenStack.
- **Tenant:** Es como se llaman en OpenStack a los proyectos. Cada uno tiene unos recursos concretos asignados, como la cantidad de instancias que se pueden crear, la cantidad de memoria que puede usar, etc.
- **Rol:** Es la función que tiene cada usuario. Un rol engloba los privilegios que tiene un usuario sobre un tenant concreto. Por ejemplo, un usuario puede tener un rol sin privilegios en un proyecto y un rol con privilegios de administrador en otro.

2.3.2.2. Servicio de imagen (Glance)

Glance [27] es el servicio encargado de gestionar las imágenes de máquinas virtuales, las cuales son necesarias a la hora de crear las instancias. Para entender correctamente el funcionamiento de este servicio hace falta definir algunos conceptos:

- **Imagen:** Es la copia de una máquina virtual y se usa para crear instancias. Contiene su sistema operativo, sus ficheros de datos y sus aplicaciones.
- **Snapshot:** Es la copia de una imagen en un momento concreto de su ejecución, también se usan para crear instancias. Por lo tanto, son un tipo especial de imágenes que además de contener el OS, los datos y las aplicaciones, contienen el estado en el que es estaba la VM en el momento de hacer el snapshot.

2.3.2.3. Servicio de cómputo (Nova)

Nova [28] es el servicio principal que ejecuta el nodo compute. Es el encargado de la gestión de las máquinas virtuales y se encarga principalmente de realizar todas las acciones relacionadas con el ciclo de vida de las instancias: iniciarlas, apagarlas, suspenderlas y reiniciarlas.

2.3.2.4. Servicio de red (Neutron)

Neutron [29] es el servicio encargado de proporcionar conectividad de red a las interfaces virtuales de las instancias creadas con OpenStack. Se encarga de asignar direcciones IP, la creación de redes y la gestión de los puertos. También gestiona los grupos de seguridad, que sirven para controlar las conexiones tanto entrantes como salientes de cada instancia, funcionando a modo de firewall.

2.3.2.5. Dashboard (Horizon)

Horizon [30] es el nombre que recibe la interfaz web de OpenStack, la cual representa gráficamente en tiempo real toda la información que contiene la plataforma. Desde Horizon los usuarios y administradores pueden acceder fácilmente al resto de los servicios, como por ejemplo gestionar los proyectos mediante Keystone, pausar una instancia mediante Nova, etc.

Capítulo 3

Montaje de la arquitectura

En este capítulo se describen los procedimientos de creación de las plataformas que forman parte de la arquitectura necesaria para el proyecto, que son OSM y OpenStack. De cada plataforma se incluyen los requisitos hardware que precisan, los valores concretos que se han usado en el montaje de este estudio y el procedimiento de instalación. En último lugar se indica cómo integrar ambas plataformas para que cooperen y así conseguir la arquitectura MANO necesaria para llegar a la solución.

3.1. Aclaraciones iniciales

Antes de leer los procedimientos necesarios para desarrollar la solución técnica, es importante tener en cuenta una serie convenciones que se usan a lo largo de este capítulo.

3.1.1. Comandos en consola

Tanto en este capítulo como en el Capítulo 7, que es referenciado en más de una ocasión, es frecuente encontrar fragmentos de código que forman parte del procedimiento para llegar a la solución. Se ha elegido el siguiente formato para representar dichos fragmentos de código:

```
1 host$       comandoX
2 controller# comandoY
3 compute$    comandoZ
```

Explicación de los elementos que forman cada sentencia:

- **Número de línea:** Aparece en la izquierda de cada sentencia sobre un fondo más oscuro. Su función es facilitar la referencia a una línea concreta en caso de ser necesario.
- **Prompt[9]:** Aparece en color azul. La razón de incluirlo en cada sentencia es facilitar al lector el seguimiento del proceso. No se muestra el contenido completo del prompt porque sería demasiado largo y quitaría protagonismo al comando. En este prompt modificado se muestran únicamente:
  - **Nombre del equipo**: Indica el equipo sobre el que se está ejecutando el comando. Se pueden encontrar cuatro nombres de equipos distintos a lo largo de este proyecto.
    - **host:** Hace referencia al equipo físico en el que se ha realizado el proyecto.
    - **osm:** Máquina virtual que tiene instalado el software de OSM.
    - **controller:** VM que tiene instalado el nodo controller de OpenStack.
    - **compute:** Máquina virtual que tiene instalado el software del nodo compute de OpenStack.
  - **Carácter especial**:
    - $: Sirve para indicar que el comando lo puede ejecutar cualquier usuario.
    - #: Significa que solo puede ejecutar el comando el administrador, es decir un usuario con privilegios de root.
- **Comando:** Es la parte de cada sentencia que contiene la información relevante.

[9] Conjunto de caracteres que se muestran en una línea de comandos para indicar que está a la espera de órdenes.

3.1.2. Ficheros de texto

Parte del procedimiento para llegar a la solución consiste en modificar algunos ficheros de texto, los cuales forman parte de la configuración de alguno de los softwares involucrados en el proyecto. Se ha elegido el siguiente formato para representar los cambios que hay que hacer sobre estos ficheros de texto:

/ruta/nombre_fichero.txt	
1	Sentencia_borrar
2	Sentencia introducir
3	Sentencia no modificar

Explicación del formato elegido:

- **Nombre del fichero:** Aparece siempre en la parte superior y se representa con el color azul.
- **Número de línea:** Aparece a la izquierda de la sentencia. Su función principal es facilitar la referencia a una línea concreta.
- **Sentencias:** Es la parte del fichero que contiene la información. A continuación, se explica la convención de colores seguida:
  - Rojo: Línea que formaba parte del fichero y es inservible, por lo que hay que eliminarla.
  - Verde: Línea nueva que hay que introducir en el fichero.
  - Negro: Línea que formaba parte del fichero y no hay que hacer ninguna modificación sobre ella.

3.2. Máquinas virtuales

Antes de instalar las plataformas necesarias para llevar a cabo este estudio es importante definir cómo van a estar implementadas sobre el equipo utilizado para realizar el estudio.

Ambas plataformas se pueden instalar directamente sobre un equipo o en una máquina virtual [31]. Se ha optado por la segunda opción por los siguientes motivos:

- **Minimizar el riesgo:** OSM es un software testeado y fiable, no obstante, aún está en fase de desarrollo por lo que es más seguro no probarlo directamente sobre un equipo. OpenStack es una tecnología mucho más madura, por lo que este motivo no le afecta directamente.
- **Seguridad:** Posibilidad de hacer copias de seguridad fácilmente, así en caso de aplicar una configuración errónea, se podría volver a una versión funcional anterior.
- **Portabilidad:** El proyecto podría exportarse a otros equipos de una manera sencilla.

Una vez establecido que se van a usar máquinas virtuales para llevar a cabo el proyecto, es necesario usar una herramienta de virtualización para poder crearlas. Existen

varias alternativas en el mercado como VirtualBox, KVM o VMWare, por lo que es necesario hacer un análisis de cuál es la más recomendable para realizar este estudio. A continuación, se exponen los motivos por los cuales se ha elegido KVM [32]:

- **Económico:** Se trata de una aplicación de código abierto y por lo tanto gratuita, al contrario de otras plataformas de virtualización, como es el caso de VMWare, que hay que pagar para poder usarlas.
- **Rendimiento:** Para crear máquinas virtuales Linux el rendimiento de KVM es superior a la de otras herramientas.

A pesar de que es fundamental para poder continuar con el proyecto, el procedimiento de instalación de KVM no está suficientemente relacionado con el tema principal del estudio como para aparecer en este capítulo. No obstante, se puede encontrar en el Anexo A una breve guía de como instalar dicha herramienta.

3.3. OSM

En este apartado se describe el procedimiento para realizar la instalación de la tecnología principal de este proyecto, OSM. Como ya se comentó anteriormente, OSM cuenta con varias versiones de software, siendo la versión FOUR la más actual y la que se ha decidido usar para el estudio.

En un principio se comenzó a trabajar sobre la versión THREE para la realización de este TFG, debido a que la FOUR todavía no existía. Sin embargo, tras la salida de la nueva versión se decidió enfocar el trabajo en ella, ya que merecía la pena probar las mejoras introducidas en la plataforma y así poder hacer un estudio más completo y actualizado de OSM.

Como se ha trabajado sobre dos versiones distintas, en ocasiones aparecen alusiones a la más antigua con el fin de mostrar lo que ha mejorado la plataforma en los seis meses de diferencia que hay entre ellas. No obstante, hay que tener en cuenta que todo el contenido de este apartado es exclusivamente aplicable a la versión FOUR, ya que los cambios producidos entre ella y las anteriores versiones son bastante notables.

3.3.1. Requisitos de OSM

En anteriores versiones los requisitos de instalación eran muy elevados, sin embargo, para la versión FOUR el grupo de trabajo de OSM hizo un gran esfuerzo para conseguir la misma funcionalidad, e incluso mejorarla en algunos aspectos, reduciendo dichos requisitos.

A continuación, se enuncian las características de las que debe disponer un equipo, o una máquina virtual, para poder instalar OSM versión FOUR [33]. Se ha establecido una comparación directa con los requisitos de la versión anterior [34] para poder apreciar fácilmente lo mucho que ha mejorado la plataforma en este aspecto:

- **2 CPUs:** Se recomienda el uso de 2 CPUs para la OSM FOUR, nada que ver con los 8 CPUs recomendados para la versión anterior.

- **8 GB de memoria RAM:** La página web de OSM recomienda 8 GB de memoria RAM. En la versión TRHEE eran necesarios 16 GB.
- **40 GB de almacenamiento:** Para la nueva versión son imprescindibles 40 GB de almacenamiento, que son la mitad de los 80 GB recomendados para el software anterior.
- **Ubuntu 16.04 (variante de *64-bit*[10]):** OSM recomienda únicamente la versión 16.04 del sistema operativo Ubuntu como imagen base para su software. No garantiza que la instalación sea posible con otras versiones de dicho sistema operativo, ya sean más antiguas o más recientes, ni tampoco con la variante de 32-bit.
- **Una interfaz de red:** Es imprescindible que la interfaz tenga acceso a internet, ya que se descargan varios GB de datos de repositorios externos.

3.3.2. Máquina virtual de OSM

Como ya se mencionó en el apartado 3.3.1. los requisitos de OSM a partir de la versión FOUR son inferiores a los de anteriores versiones, por lo que es perfectamente plausible crear una máquina virtual que cumpla con las especificaciones recomendadas. Esto no era posible con la versión THREE, ya que sus necesidades eran superiores a lo que el equipo sobre el que se está realizando el TFG podía soportar y por lo tanto había que adaptarlas.

Se procede a crear una VM que cumple con los requisitos recomendados especificados en el apartado 3.3.1., es decir:

- **Número de CPUs**: 2 CPUs.
- **Memoria RAM**: 8 GB.
- **Disco duro**: 40 GB.
- **Sistema Operativo**: Ubuntu 16.04 (64-bit).

El procedimiento de creación de una máquina virtual en KVM no es objeto de este estudio, por lo que no aparece en este capítulo. Pero, como sí es imprescindible para poder llegar a la solución, se pueden encontrar los pasos necesarios en el Anexo B.

Una vez creada la máquina virtual hay que instalarle el sistema operativo, que nuevamente no aparece en este capítulo por los mismos motivos proporcionados en párrafo anterior. En el Anexo C se puede encontrar una guía de cómo hacerlo.

3.3.3. Instalación de OSM

El procedimiento de instalación de OSM [33] es otro de los aspectos que han sido mejorados en la versión FOUR. Anteriormente había que hacer una serie de configuraciones previas para poder comenzar con la instalación de la plataforma. Esos

[10] Actualmente existen dos tipos de arquitecturas informáticas: 32-bit y 64-bit; las cuales hacen referencian a cómo los equipos informáticos almacenan los datos. Cuanto mayor sea el número más información pueden manejar los equipos en menos tiempo.

pasos previos han sido automatizados, por lo que ahora la instalación de OSM es mucho más sencilla.

```
1 host$  wget https://osm-download.etsi.org/ftp/osm-4.0-four/install_osm.sh
2 host#  chmod +x install_osm.sh
3 host#  ./install_osm.sh
```

- **Comando 1:** Descargar el *script*[11] de instalación de OSM.
- **Comando 2:** Otorgar permisos de ejecución al script.
- **Comando 3:** Ejecutar el script de instalación.

El script de instalación contiene cientos de sentencias que se van ejecutando automáticamente y evita al usuario tener que hacerlo de manera manual. A los pocos minutos de ejecutar el último comando hay que responder a las siguientes preguntas, que sirven para configurar los contenedores sobre los que va a estar implementada parte de la arquitectura de OSM.

- Do you want to configure a new storage pool (yes/no) [default=yes]?
  - **Respuesta:** yes.
  - **Explicación:** Los contenedores necesitan espacio de almacenamiento por lo que hay que proporcionárselo.
- Name of the storage backend to use (dir or zfs) [default=dir]:
  - **Respuesta:** dir.
  - **Explicación:** Cualquiera de las dos opciones es válida. Elegir "zfs" permite gestionar detalladamente el almacenamiento de los contenedores, pero como no es imprescindible, es preferible seleccionar la opción "dir", la cual no precisa de configuración adicional.
- Would you like LXD to be available over the network (yes/no) [default=no]?
  - **Respuesta:** no.
  - **Explicación:** Los contenedores no tienen que ser accesibles desde internet.
- Do you want to configure the LXD bridge (yes/no) [default=yes]?
  - **Respuesta:** yes.
  - **Explicación:** Es imprescindible la creación y configuración del "LXD bridge", ya que es lo que permite la conectividad entre los contenedores y el equipo host.

Inmediatamente después de responder a las preguntas anteriores aparece un *wizard*[12] de instalación que permite seguir configurando el LXD bridge. Para facilitar el seguimiento de este proceso se muestran las capturas de las respuestas que hay que indicar para que la instalación se haga de manera correcta. El procedimiento es el siguiente:

- **Fig. 3.1:** Confirmación de la creación del bridge LXD.

[11] En informática, un script, es un conjunto de órdenes guardadas en un archivo de texto. Al usar el script se ejecutan las ordenes contenidas de manera automática.
[12] Asistente de instalación.

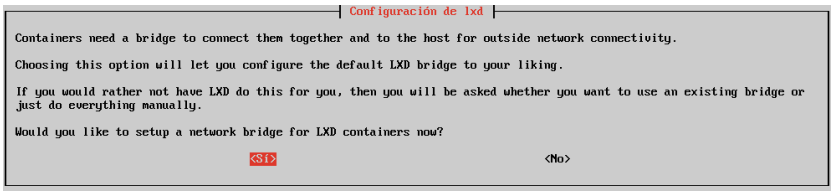

Fig. 3.1. Confirmación de creación del bridge LXD

- **Fig. 3.2:** Elección del nombre para el bridge.

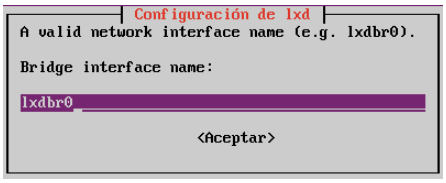

Fig. 3.2. Asignación del nombre del bridge LXD

- **Fig. 3.3:** Configuración del bridge para tener conectividad IPv4.

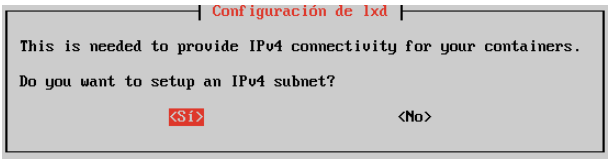

Fig. 3.3. Creación de subred IPv4

- **Fig. 3.4:** Asignación de IPv4 al bridge.

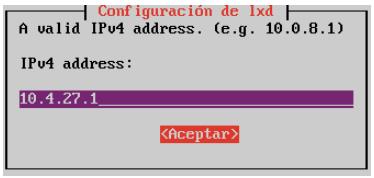

Fig. 3.4. Asignación de IPv4 al bridge

- **Fig. 3.5:** Asignación de máscara de subred.

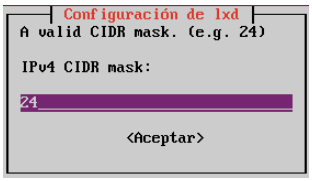

Fig. 3.5. Configuración de máscara de subred del bridge

- **Fig. 3.6 y Fig. 3.7:** Elección de la primera y la última dirección del pool *DHCP*[13], el cual es necesario para que los contendores obtengan sus direcciones IP.

Fig. 3.6. Primera dirección del pool DHCP

[13] Abreviación de Dynamic Host Configuration Protocol. Es un protocolo utilizado para asignar dinámicamente direcciones IP y así los equipos puedan comunicarse con otras redes.

Fig. 3.7. Última dirección del pool DHCP

- **Fig. 3.8:** No hay que configurar el bridge para tener conectividad IPv6, ya que no es necesaria.

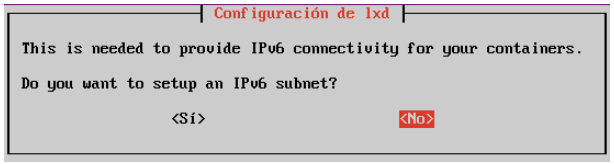

Fig. 3.8. Creación de subred IPv6

Tras finalizar la última pregunta del wizard, continúa la ejecución del script, la cual dura una hora aproximadamente, aunque puede variar dependiendo de la conexión a internet.

Cuando acabe la ejecución del script significa que la instalación de OSM ha concluido, no obstante, hay que hacer un último paso antes poder usar la plataforma y es incluir dos *variables de entorno*[14] al final del fichero ".bashrc". Incluir las variables en el fichero evita tener que crearlas de nuevo cada vez que se reinicie el equipo, ya que el fichero ".bashrc" se ejecuta siempre en el arranque del sistema y por lo tanto se crean automáticamente.

```
1 osm$ echo "export OSM_HOSTNAME=127.0.0.1" >> .bashrc
2 osm$ echo export OSM_SOL005=True "" >> .bashrc
```

- **Comando 1:** Añade la variable de entorno "OSM_HOSTNAME", esto implica que cuando se referencie a dicha variable sea lo mismo que indicar la dirección IP de localhost.
- **Comando 2:** Añade la variable de entorno "SOL005", esto indica que la NBI, explicada en el apartado 2.3.1.9., está activa y por lo tanto los componentes de la arquitectura de OSM que la necesiten sepan que pueden utilizarla.

3.3.4. Acceso a Launchpad

Para comprobar que la instalación de OSM server ha concluido satisfactoriamente lo más sencillo es intentar acceder a Launchapad, la Graphical User Interface (GUI) de OSM. Para ello es necesario conocer la dirección IP de la máquina virtual sobre la cual se ha instalado OSM.

```
1 osm$ ifconfig
```

[14] Una variable de entorno es un nombre asociado a una cadena de caracteres. Las usan los programas o los propios usuarios para no tener que recordar cadenas de caracteres complejas.

- **Comando:** Mostrar las direcciones de todas las interfaces de la VM, tanto físicas como virtuales.

En el caso de la implementación realizada para este TFG la interfaz buscada se llama ens3, cuya dirección IP es 192.168.122.21. Basta con poner esa dirección en un navegador web para acceder a Lauchpad: http://192.168.122.21.

Si tras introducir la dirección IP en el navegador aparece la página de acceso a la plataforma, mostrada en la Fig. 3.9, significa que la instalación ha sido exitosa. En caso contrario pueden estar pasando dos cosas:

- **Falta de conectividad con la máquina virtual:** Si se usa KVM, la máquina virtual y el equipo host pertenecen a la misma subred, por lo que para acceder a Launchpad no se necesita configuración adicional. Este comportamiento no se puede garantizar en otras herramientas de virtualización, por ejemplo, VirtualBox, en su configuración por defecto, accede a las máquinas virtuales a través de un NAT por lo que una ruta directa no funcionaría. La solución a este problema consiste en crear una ruta hacia Launchpad.

```
1 host#  route add <ip_lauchpad> gw <puerta_enlace>
```

  - **Comando:** Creación de ruta en Linux, ejemplo genérico.
- **Fallo en la instalación:** Si la solución propuesta en el punto anterior no funciona significa que se ha producido un fallo durante la instalación. Cuando se ejecuta el script de instalación se descargan varios GB de repositorios externos por lo que una mala conexión a internet puede suponer que falle el proceso. La solución a este problema es ejecutar de nuevo el script de instalación.

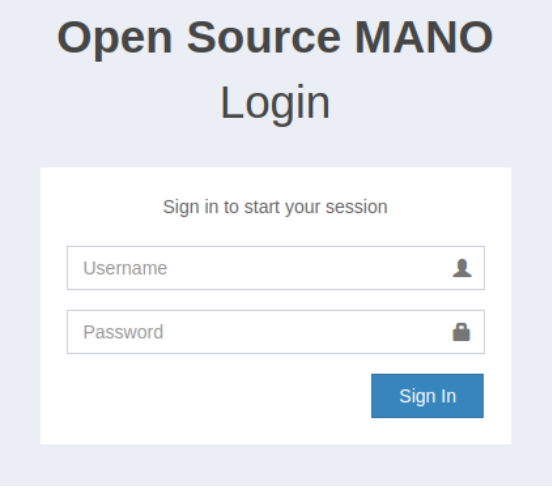

Fig. 3.9. Acceso a Launchpad

Una vez que se tenga acceso a la plataforma basta con autenticarse con el usuario y contraseña por defecto:

- **Usuario:** admin
- **Password:** admin

3.4. OpenStack

En este apartado se explicará el procedimiento de instalación de OpenStack, plataforma que hace las funciones del módulo VIM en este proyecto.

Existen dos alternativas a la hora de implementar OpenStack:

- **Devstack** [35]**:** Es un script que sirve para implementar un entorno completo de OpenStack de manera automática. Sus principales ventajas son:
  - Comodidad.
  - Rapidez.
- **Manual:** Consiste en instalar y configurar los módulos de OpenStack a mano. Sus principales ventajas son:
  - Capacidad de decisión a la hora de elegir los servicios a implementar.
  - Menor uso de los recursos hardware.

Aunque lo más cómodo y rápido sea obtener la plataforma mediante la ejecución del script Devstack, se ha optado por hacer la instalación de manera manual. El motivo principal de esta decisión es que para implementar el módulo VIM son únicamente necesarios los servicios comentados en el apartado 2.3.2. La implementación con esos servicios únicamente se puede conseguir mediante la instalación manual, por lo que si se utilizara Devstack se crearían más servicios de los imprescindibles y no se haría un uso eficiente de los recursos hardware, los cuales son muy limitados.

3.4.1. Requisitos de OpenStack

La arquitectura de OpenStack no está predefinida, ya que se pueden elegir los nodos que se van a implementar, eso implica que los requisitos varían según el número final de nodos instalados.

Para este TFG se van a implementar únicamente los nodos imprescindibles en una arquitectura OpenStack: el nodo compute y el nodo controller. Cada uno de ellos desempeña funciones muy distintas dentro de la plataforma, por lo que es normal que los requisitos que precisan sean diferentes.

3.4.1.1. Requisitos de instalación del nodo controller

Para poder instalar el nodo controller se necesita un equipo, o máquina virtual, con las siguientes características [24]:

- **1 o más CPUs:** OpenStack recomienda el uso de 2 CPUs, no obstante, también ofrece la posibilidad de hacer una instalación más ligera con únicamente 1 CPU.
- **4 GB o más de memoria RAM:** Se establece el mínimo de 4 GB de memoria RAM para implementar el nodo controller, aunque lo más aconsejable sea hacer una implantación con al menos 8 GB.
- **50 GB o más de memoria de almacenamiento:** Se recomienda 100 GB de memoria de almacenamiento, pero con 40 GB es suficiente.

- **Linux (variante de 64-bit):** La versión 16.04 del sistema operativo Ubuntu es la más recomendada como imagen base para instalar posteriormente el nodo controller de OpenStack. Aunque cualquier otra distribución Linux es válida, siempre que sea la variante de 64-bit.

3.4.1.2. Requisitos de instalación del nodo compute

Para poder instalar el nodo compute se necesita un equipo o una máquina virtual con las siguientes características [24]:

- **2 o más CPUs:** Lo más recomendable es usar 4, no obstante, es posible usar únicamente 2 CPUs.
- **8 GB o más de memoria RAM:** OpenStack recomienda al menos 8 GB de memoria RAM.
- **100 GB o más de memoria de almacenamiento:** Se establecen los 100 GB de memoria de almacenamiento como la opción más recomendable.
- **Linux (variante de 64-bit):** De nuevo la versión 16.04 del sistema operativo Ubuntu es imagen base más recomendada. Aunque otra variante 64-bit de cualquier distribución Linux también es válida.

3.4.2. Creación de las máquinas virtuales para OpenStack

Hay que crear una máquina virtual para alojar al nodo controller que cumpla con los requisitos mínimos especificados en el apartado 3.4.1.1., es decir:

- **Número de procesadores:** 1 CPUs.
- **Memoria RAM:** 4 GB.
- **Disco duro:** 40 GB.
- **Sistema Operativo:** Ubuntu 16.04 (64-bit).

Y otra máquina virtual para el nodo compute que cumple las especificaciones recomendadas del apartado 3.4.1.2., como son:

- **Número de procesadores:** 2 CPUs.
- **Memoria RAM:** 8 GB.
- **Disco duro:** 100 GB.
- **Sistema Operativo**: Ubuntu 16.04 (64-bit).

Como se puede observar las especificaciones hardware del nodo compute son bastante superiores a los del nodo controller, incluso más del doble en algunos aspectos. Esto se debe a que el nodo compute es donde se van a crear las instancias, es decir, todos los recursos de este nodo son en realidad los recursos de los que van a disponer las futuras VNFs. Esto supone que las especificaciones que se le otorguen a dicho nodo van a determinar el número de instancias, o la complejidad de las mismas, que se puedan crear posteriormente.

En el Anexo B se encuentra el procedimiento para crear las dos máquinas virtuales necesarias, a las cuales hay que instalarles el sistema operativo, explicado en el Anexo C.

3.4.3. Procedimiento de instalación de OpenStack

OpenStack, el módulo VIM de este proyecto, es un componente fundamental y de gran relevancia dentro de la arquitectura MANO que se pretende crear. El procedimiento de instalación de la plataforma es bastante extenso y no está lo suficientemente relacionado con el objeto de estudio de este TFG, por lo que no tiene cabida en este capítulo. No obstante, se ha incluido una guía del procedimiento de instalación en el Anexo D.

En la *Fig. 3.10*[15], se muestra la arquitectura obtenida, con los valores concretos utilizados para este proyecto, tras haber realizado todo el procedimiento descrito en el Anexo D.

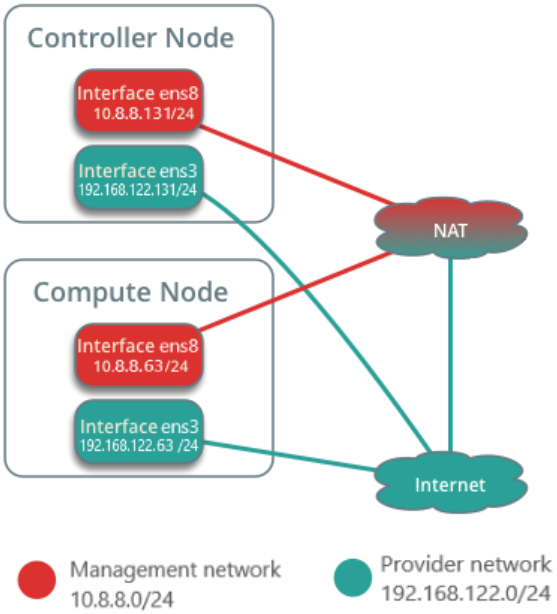

Fig. 3.10. Arquitectura OpenStack del proyecto [36]

3.5. Integración de OSM y OpenStack

Una vez instalados OSM y OpenStack, se disponen de todos los componentes necesarios de una arquitectura MANO completa, no obstante, ambas plataformas siguen funcionando de manera independiente por lo que hay que integrarlas.

3.5.1. Configuración de OpenStack como VIM

Antes de poder hacer la integración de las dos plataformas hay que configurar OpenStack para que pueda ser usado por OSM como módulo VIM, como se indica en [37].

[15] Imagen modificada para ajustarse a la implementación del proyecto. La imagen está sujeta a la licencia Creative Commons que permite dicha modificación.

3.5.1.1. Creación de las redes virtuales

Para que en el futuro se puedan crear instancias en la plataforma hay que crear primero una infraestructura de redes virtuales que puedan contenerlas. Las instancias no pueden estar "flotando" sin más, necesitan una red que les proporcione direccionamiento, acceso a internet, etc.

En OpenStack se pueden crear tantas redes virtuales como se requieran, sin embargo, hay una de ellas que debe crearse siempre en primer lugar ya que es la que sirve de sustento a las demás. Esta red principal, que suele recibir el nombre de "provider", es la representación virtual de la red física donde se encuentra implementada la plataforma, por lo tanto, en ella se encuentran los nodos controller y compute de OpenStack, además del gateway. El procedimiento para su creación es el siguiente:

```
1 controller#  . admin-openrc
2 controller#  openstack network create --share --external --provider-physical-network provider --
               provider-network-type flat provider
3 controller#  openstack subnet create --network provider --allocation-pool
               start=192.168.122.140,end=192.168.122.240 --dns-nameserver 8.8.8.8 --gateway
               192.168.122.1 --subnet-range 192.168.122.0/24 provider
```

- **Comando 1:** Cargar las credenciales del usuario "admin" para tener los privilegios necesario para poder ejecutar el resto de los comandos.
- **Comando 2:** Crear la red virtual.
- **Comando 3:** Crear la subred virtual. Este comando incluye varios elementos importantes:
  - **Direccionamiento:** Tiene que coincidir con el de la red física, en este caso 192.168.122.0/24.
  - **Pool de direcciones:** Esta nueva red necesita un servidor DHCP para asignarle direcciones IP a las futuras instancias. Es muy importante que en el pool de direcciones no estén incluidas las direcciones que ya estén en uso (controller: 192.168.122.131, compute: 192.168.122.63 y osm-server: 192.168.122.34) ya que sí se le asigna la misma dirección a una instancia habría un problema de IPs duplicadas.
  - **Servidores *DNS*[16]:** No es obligatorio, pero sí muy recomendable, especificar un servidor DNS en la subred.
  - **Gateway:** Es imprescindible especificar el gateway si se quiere que las instancias tengan conectividad con Internet.

Una vez creada la red "provider" ya se podrían crear el resto de las redes virtuales que fueran necesarias. Para los escenarios que se han implementado en este TFG no es

[16] DNS: Acrónimo de Domain Name System, es sistema de nomenclatura cuya función más importante es traducir nombres de domino a direcciones IP, las cuales son difíciles de recordar para las personas. Por ejemplo, es más sencillo recordar el nombre de dominio "uc3m.es" que la dirección IP del servidor que proporciona la página web de la universidad.

necesario crear una infraestructura de red más compleja, es decir, solo se va a usar la red "provider" y por lo tanto las instancias se sitúan directamente sobre esta red.

3.5.1.2. Creación de usuario y proyecto

Para que OSM pueda crear instancias en OpenStack necesita tener los privilegios adecuados. Esto se puede conseguir creando un nuevo usuario y otorgándole los privilegios del rol administrador "admin". Además, es conveniente crear un nuevo tenant para todo lo relacionado con OSM, ya que la forma correcta de usar OpenStack, es utilizar proyectos independientes para agrupar las implementaciones que tengan un mismo propósito. El procedimiento de creación del nuevo usuario es el siguiente:

```
1 controller#  openstack project create --description 'Proyecto de osm' osm-project --domain default
2 controller#  openstack user create --project osm-project --password osm.1234 osm
3 controller#  openstack role add --user osm --project osm-project admin
```

- **Comando 1:** Crear el proyecto "oms-project", en el que se van a crear todas las instancias relacionadas con OSM.
- **Comando 2:** Crear el usuario "osm", que tiene la contraseña "osm.1234" asociada.
- **Comando 3:** Asignar el rol "admin" al nuevo usuario "osm".

3.5.1.3. Configuración de los grupos de seguridad

Por defecto OpenStack aplica a todas las instancias que se creen el grupo de seguridad por defecto "default", que bloquea todo el tráfico entrante a la máquina virtual. Sin embargo, para que OSM pueda usar a OpenStack como módulo VIM, necesita poder acceder a él mediante el protocolo[17] *SSH*[18]. Hay dos posibles soluciones a este problema:

- **Solución 1:** Crear un nuevo grupo de seguridad.
- **Solución 2:** Modificar el grupo de seguridad "default".

Para el uso que se hace de OpenStack en este TFG, que es usar un único proyecto y crear allí todas las instancias relacionadas con OSM, lo más recomendable es modificar el grupo de seguridad "default", ya que cambiar este grupo no va a afectar a más proyectos. Otro escenario totalmente distinto sería si hubiese varios proyectos en OpenStack, ya que alguno de ellos podría estar usando el grupo de seguridad "default" y por lo tanto modificarlo alteraría también a dichos proyectos.

```
1 controller#  openstack security group rule create --proto tcp --dst-port 22 default
```

- **Comando:** Creación de una nueva regla, en el grupo de seguridad "default", que permite el tráfico entrante por el puerto 22, que es el del protocolo SSH.

[17] Es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de la red.
[18] SSH, es la abreviación de Secure Shell. Es un protocolo que permite la comunicación segura entre dos equipos.

3.5.2. Integración de las plataformas

Una vez preparado OpenStack para hacer las funciones del módulo VIM se puede integrar dicha plataforma en la arquitectura de OSM [33].

```
1  osm$  osm vim-create --name openstack-site --user osm --password osm.1234 --auth_url
         http://10.8.8.131:5000/v3 --tenant osm-project --account_type openstack --
         config='{security_groups: default, use_floating_ip:true}'
```

- **Comando**: Creación de un nuevo VIM en la plataforma OSM. Para ello se indican los siguientes elementos:
  - **Tipo de módulo VIM**: OpenStack es la plataforma que hace las funciones de módulo VIM en esta implementación.
  - ***URL*[19] de autenticación:** URL para acceder al módulo VIM, en este caso la dirección del servicio de identidad de OpenStack (Keystone), alojado en el nodo controller.
  - **Usuario y contraseña:** Nombre del usuario que usa OSM para identificarse en el módulo VIM, en este caso el usuario "osm" creado en el apartado 3.5.1.2.
  - **Projecto:** Nombre del proyecto en el cual va a trabajar OSM, en este caso el proyecto "osm-project" creado en el apartado 3.5.1.2.
  - **Grupo de seguridad:** Nombre del grupo de seguridad que se debe aplicar a las instancias creadas desde OSM, en este caso el grupo por defecto "default", modificado en el apartado 3.5.1.3.

Tras la integración se obtiene la arquitectura de la Fig. 3.11, en la cual se muestran los valores concretos usados en la implementación de este proyecto.

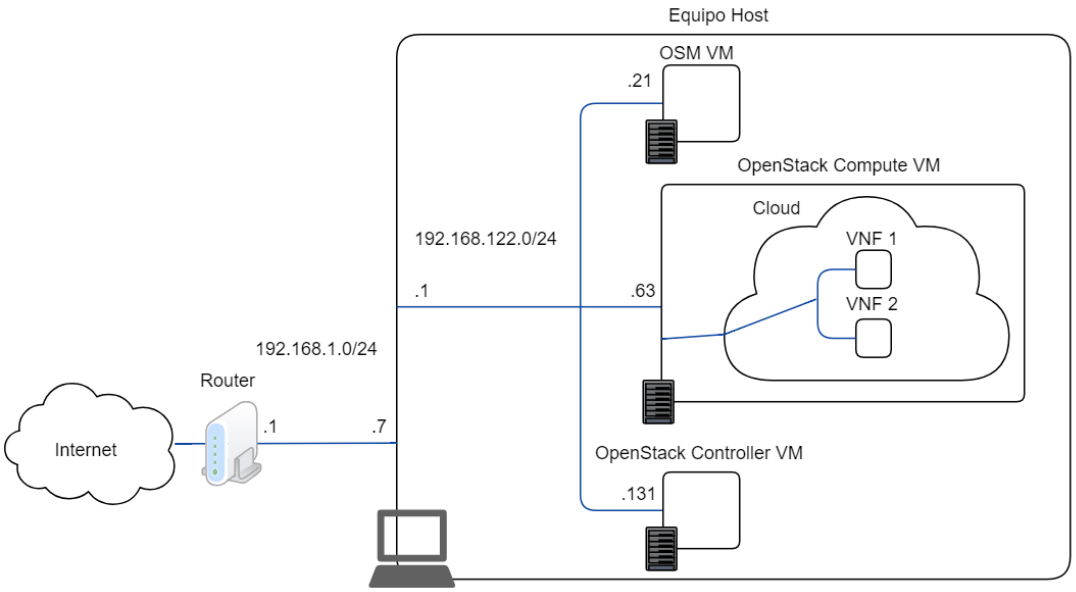

Fig. 3.11. Arquitectura completa

[19] URL es la abreviación de Uniform Resource Locator. Es una secuencia de caracteres que permite identificar recursos dentro internet para que puedan ser localizados.

Capítulo 4

Creación de Network Services y evaluación de resultados

En este capítulo se diseñan diferentes Network Services para evaluar la plataforma de orquestación NFV OSM. Desplegar estos NS permite, en primer lugar, verificar la correcta instalación e integración de todos los componentes de la arquitectura implementada y, posteriormente, evaluar el potencial del orquestador, el cual es el objetivo principal de este estudio. Una vez determinado el potencial se puede realizar una evaluación de OSM, que está centrada en un análisis del rendimiento de la plataforma. También se prueban funcionalidades poco comunes que incorpora OSM y lo diferencian del resto de orquestadores.

4.1. Network Services

En este apartado se evaluarán tres NS: uno muy sencillo proporcionado por OSM y los otros diseñados expresamente para poder estudiar en más profundidad el orquestador.

4.1.1. Network Service 1: Escenario de prueba

OSM proporciona un NS listo para ser instanciado a modo de ejemplo, el cual es muy sencillo y por lo tanto no permite evaluar el potencial del orquestador. Sin embargo, es excelente como toma de contacto con la plataforma, además permite comprobar que la instalación de los módulos de OSM se ha realizado correctamente.

El NS, representado en la Fig. 4.1, está formado por dos VNFs con sistema operativo base *CirrOS*[20] unidos por un Virtual Link.

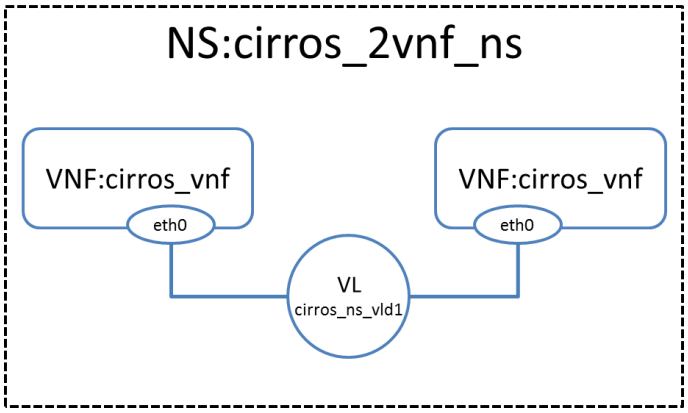

Fig. 4.1. Topología Network Service 1 [38]

4.1.1.1. Instanciación

El procedimiento para desplegar el NS a través de OSM [33] es el siguiente:

- **Añadir la imagen a Openstack:** En primer lugar, hay que incorporar la imagen que van a usar las futuras instancias a OpenStack, en este caso la imagen CirrOS.

```
1  controller#  wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
2  controller#  openstack image create --file="./cirros-0.3.4-x86_64-disk.img" --container-
                format=bare --disk-format=qcow2 --public cirros034
```

  - **Comando 1:** Descargar la imagen.
  - **Comando 2:** Incorporar CirrOS al catálogo de imágenes disponibles de OpenStack, administrado por el servicio Glance.
- **Incorporación de los descriptores:** Para poder desplegar el NS hay que incluir los descriptores en OSM. Siempre se debe incorporar primero el VNFD, ya que un atributo del NSD lo referencia, por lo tanto, en caso de hacerlo al revés se obtendría un error.

[20] Sistema operativo ligero específicamente diseñado para funcionar en entornos cloud.

```
1 osm#  wget https://osm-download.etsi.org/ftp/osm-3.0-
        three/examples/cirros_2vnf_ns/cirros_vnf.tar.gz
2 osm#  osm upload-package cirros_vnf.tar.gz
3 osm#  wget https://osm-download.etsi.org/ftp/osm-3.0-
        three/examples/cirros_2vnf_ns/cirros_2vnf_ns.tar.gz
4 osm#  osm upload-package cirros_2vnf_ns.tar.gz
```

- **Comando 1:** Descargar el VNFD del servidor FTP de OSM.
- **Comando 2:** Incluir el nuevo descriptor en el catálogo de VNFDs disponibles de OSM.
- **Comando 3:** Descargar el NSD del servidor FTP de OSM.
- **Comando 4:** Incluir el descriptor en el catálogo de NSDs de OSM.

- **Instanciación del NS:** Una vez que estén todos los descriptores incorporados a la plataforma se puede crear el servicio de red.

```
1 osm#  osm ns-create --nsd_name cirros_2vnf_ns --ns_name test --vim_account openstack-
        site
```

- **Comando:** Crear un nuevo servicio de red llamado “test” usando como módulo VIM “openstack-site”.

4.1.1.2. Evaluación del Network Service

En este apartado se realizan una serie de pruebas al NS para verificar su correcto funcionamiento. Como el NS es sencillo, las pruebas resultaran igualmente sencillas.

4.1.1.2.1. Prueba 1: Creación

La primera prueba consiste en comprobar que la comunicación entre OSM y el módulo VIM funciona correctamente, es decir, que la petición de creación de las instancias realizada por OSM ha sido interpretada y realizada por OpenStack.

```
1 controller#  openstack server list
```

- **Comando:** Mostrar una lista de las instancias creadas.

Como se puede ver en la Fig. 4.2, hay dos instancias creadas en OpenStack, las cuales tienen las direcciones IP 192.168.122.153 y 192.168.122.143. Esta información también se puede ver a través de Horizon de una manera gráfica, como se muestra en la Fig. 4.3.

ID	Name	Status	Networks	Image Name
f7873eba-66e6-472e-9620-57c55080 8b70	NS_1.cirros_vnf. 2.cirros_vnfd-VM	ACTIVE	provider=192.168 .122.153	cirros034
d1166986-78c3 -455c-aac8-a11ab 2491b7e	NS_1.cirros_vnf. 1.cirros_vnfd-VM	ACTIVE	provider=192.168 .122.143	cirros034

Fig. 4.2. Lista de instancias

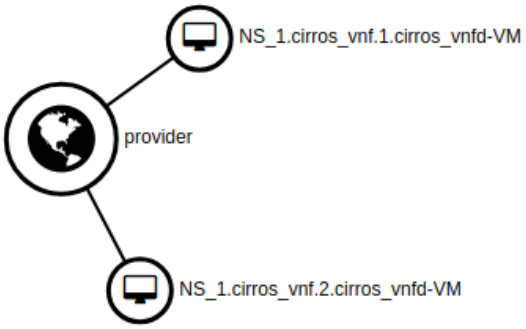

Fig. 4.3. Instancias creadas vistas desde Horizon

4.1.1.2.2. Prueba 2: Conectividad

Al haberse creado dos instancias se puede comprobar la conectividad entre ellas mediante un *ping*[21]. También es interesante probar la conexión de alguna de ellas con internet.

Las instancias creadas pertenecen al grupo de seguridad "default" que bloquea todo el tráfico entrante. Hay que añadirle una nueva regla para permitir ping, como ya se hizo para el protocolo SSH en el apartado 3.5.1.3.

```
1 controller#  openstack security group rule create --proto icmp default
```

- **Comando:** Añadir una nueva regla al grupo de seguridad "default" para permitir el tráfico *ICMP*[22], ya que el ping es un tipo de mensaje de este protocolo.

Para poder hacer el ping hay que acceder primero a las instancias, esto se puede hacer a través de la GUI de OpenStack, Horizon. Una vez dentro de las instancias, se puede probar la conectividad con los siguientes comandos:

```
1 instancia$     ping 192.168.122.153
2 instancia$     ping openstack.org
```

- **Comando 1:** Probar conectividad con la otra instancia.
- **Comando 2:** Probar la conectividad con internet. Al mismo tiempo también se está comprobando que funciona el DNS, ya que se ha realizado el ping a un servidor cuya IP es desconocida para la instancia.

4.1.1.2.3. Prueba 3: Acceso remoto

Para conseguir acceder remotamente mediante el protocolo SSH no sirven las instancias con la que se ha estado trabajando en las pruebas anteriores. Esto se debe a que

[21] Comando que permite verificar el estado de una determinada conexión.
[22] ICMP, es la abreviación de Internet Control Message Protocol. Es el protocolo de control y notificación de errores de IP.

antes de su creación hay que proporcionarles una clave SSH para que el equipo host pueda acceder a ellas.

El procedimiento es el siguiente:

- **Generar clave SSH:** Hay que generar la clave en el equipo con el que se vaya a acceder posteriormente a las instancias, en este caso el equipo host.

```
1 host#  ssh-keygen -q -N cloud.key
```

  - **Comando:** Generar clave SSH.

- **Crear instancia:** Ahora al crear las instancias hay que proporcionarles la clave SSH. En este caso no se van a crear por consola, sino a través de Launchpad, con el fin de probar otra alternativa que ofrece OSM. La Fig. 4.4 ilustra el formulario de creación.

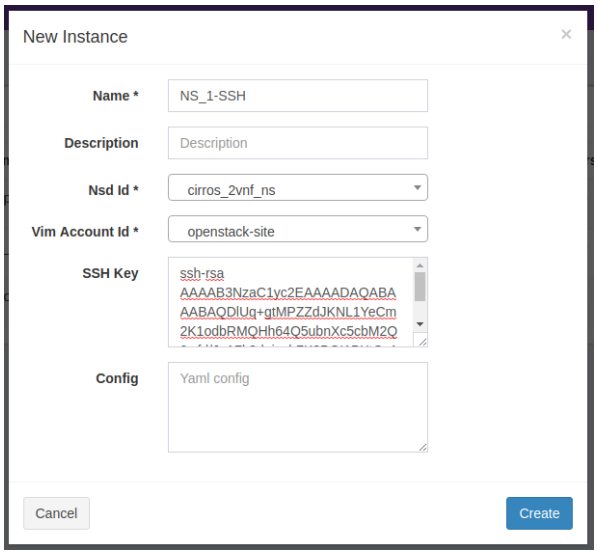

Fig. 4.4. Creación de instancias a través de Launchpad

- **Acceder a la instancia:** Como el tráfico SSH ya se permitió en el apartado 3.5.1.3. ya es posible acceder a la instancia.

```
1 host#  ssh -i cloud.key cirros@192.168.122.145
```

  - **Comando:** Acceder a la instancia mediante SSH con la clave generada. La dirección IP difiere de las pruebas anteriores, ya que las instancias se han creado de nuevo y DHCP ha asignado IPs diferentes.

4.1.2. Network Service 2: Recodificación de vídeo simple

Una vez que se ha probado el NS de ejemplo y, por lo tanto, se ha verificado que OSM funciona correctamente, se pueden instanciar escenarios más complejos, que permitirán evaluar el potencial de OSM.

El nuevo NS no existe y tiene que crearse desde cero, lo cual permite comprender mejor el funcionamiento de los descriptores y evaluar un servicio con una finalidad interesante.

El nuevo NS va a constar de dos VNFs, mostrado en la Fig. 4.5, una se encarga de la transmisión de vídeo y la otra se va a encargar de modificar dicha transmisión, es decir va a recodificar el vídeo. La recodificación puede hacerse de muchas maneras, por ejemplo: cambiar la tasa del vídeo (aumentarla o bajarla), cambiar el modo de transmisión (pasar de unicast a multicast o viceversa), etc.

Para el servidor de vídeo se ha elegido el servidor multimedia VideoLAN (VLC), que es una aplicación relativamente común que muchos usuarios conocen, aunque sea simplemente para visualizar vídeos. No obstante, se trata de una aplicación mucho más compleja que permite tanto la emisión como la recodificación del vídeo.

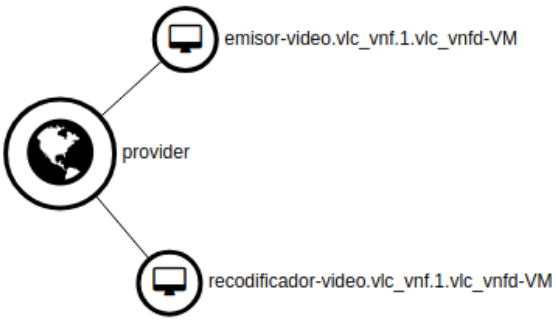

Fig. 4.5. Topología Network Service 2

4.1.2.1. Diseño

Al contrario de lo que ocurría con el NS proporcionado por OSM, ahora no se disponen de los elementos necesarios para desplegar el NS: la imagen de la instancia y los descriptores VNFD y NSD. Esto supone tener que diseñarlos y crearlos.

4.1.2.1.1. Creación de la imagen

Para desplegar un servidor de vídeo, la imagen con sistema operativo base CirrOS utilizada en el escenario anterior no sirve. Esto se debe a que dicho OS ofrece una funcionalidad muy básica, que para desplegar escenarios más complejos es insuficiente.

A la hora de seleccionar un nuevo sistema operativo para las futuras instancias hay que tener en cuenta que tiene que ser lo suficiente potente para poder ejecutar un servidor de vídeo y también que tiene que estar especialmente diseñado para ser ejecutado en entornos en la nube. Por ejemplo, la imagen Ubuntu-Server utilizada anteriormente para crear las máquinas virtuales de OSM y de OpenStack no sería válida, ya que no está optimizada para funcionar en entornos cloud, a pesar de que podría ejecutar un servidor de video sin ningún problema. Afortunadamente suele existir una versión optimizada para funcionar en la nube por cada OS. Por ello se va a usar Ubuntu-Cloud, sistema operativo que desde el punto de vista del usuario funciona exactamente igual que Ubuntu-Server, pero que puede ser instanciado correctamente en Openstack.

```
1 controller# wget https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-
              disk1.img
2 controller# md5sum xenial-server-cloudimg-arm64-disk1.img
```

- **Comando 1:** Descargar la imagen de Ubuntu-Cloud.
- **Comando 2:** Comprobar que la imagen se ha descargado correctamente, para ello hay que calcular el hash *MD5*[23] y compararlo con el proporcionado por la página web de Ubuntu. No se trata de un paso obligatorio, pero es una buena práctica realizarlo en archivos que ocupan varios MB, ya que son más propensos a errores.

Si el MD5 obtenido no coincide con el proporcionado por la web hay que descargar la imagen de nuevo. Por el contrario, si se obtiene el mismo hash, que es lo más habitual, se puede incluir la imagen a OpenStack.

```
1 controller#  openstack image create --file="./xenial-server-cloudimg-amd64-disk1.img" --container-
               format=bare --disk-format=qcow2 --public ubuntu
```

- **Comando:** Añadir la imagen de Ubuntu-Cloud, llamada "ubuntu", a OpenStack para poder crear instancias en el futuro.

A partir de este momento ya se podrían crear instancias en OpenStack a través de OSM, sin embargo, dichas instancias solo contarían con el sistema operativo sin ningún software adicional instalado. Eso implica que cada vez que se creara una instancia con esa imagen habría que instalar el servidor de video desde cero, lo cual es una pérdida de tiempo. La solución a este problema consiste en conseguir una imagen que tenga ya instalado el software del servidor de vídeo, así instanciándola se obtiene de manera automática el servidor de vídeo, sin tener que repetir configuración. Para conseguir la imagen descrita hay que partir de la imagen de Ubuntu-Cloud descargada e ir modificándola. El procedimiento es el siguiente:

- **Instanciar la imagen original:** En primer lugar, hay que instanciar la imagen de Ubuntu-Cloud a partir de Openstack.

  ```
  1 controller#  openstack server create –flavor medium --image ubuntu --nic net-id=provider --
                 security-group default --key-name mykey instancia-original
  ```

  - **Comando:** Crear una instancia con imagen "ubuntu" en la red "provider", la cual pertenece al grupo de seguridad "default" y que tiene la clave SSH "mykey".
- **Acceder a la instancia:** La única manera de acceder a la instancia es mediante SSH, para ello se proporciona la clave "mykey". No se puede acceder a través de OpenStack debido a que al arrancar la instancia esta espera que se identifique con un usuario y contraseña que no existen, por lo que es imposible acceder de esta manera sin realizar una configuración adicional. No obstante, en el apartado 4.1.2.1.2. (sección cloud-init file) se ofrece una solución a este problema, que de momento no se puede usar en este caso.

  ```
  1 host#  ssh -i cloud.key ubuntu@<ip_instancia>
  ```

[23] Acrónimo de Message-Digest Algorithm 5. Es un antiguo algoritmo de cifrado, que dejo de utilizarse para ese fin porque se detectaron problemas de seguridad. Actualmente su usa principalmente para comprobar que algún archivo no haya sido modificado.

- **Comando:** Acceder desde el equipo host a la instancia recién creada.

- **Instalar el software del servidor de vídeo:** Cuando se pueda acceder a la instancia hay que instalarle los componentes necesarios para el servidor de video.

```
instancia#  sudo apt-get update
instancia#  sudo apt-install vlc
```

  - **Comando 1:** Actualizar los repositorios para que posteriormente se descargue la versión más reciente del servidor de vídeo.
  - **Comando 2:** Instalar la aplicación VLC.

Aunque se trate de una instalación muy sencilla, ya que el proceso consta de dos comandos solamente, requiere varios minutos, por lo que tener una imagen con el servidor listo para usar, a la larga, ahorra mucho tiempo.

- **Obtener imagen modificada:** Para guardar la imagen modificada basta con crear un snapshot, que es una copia del estado de la instancia en el momento que se crea dicha réplica.

```
controller#  openstack snapshot create –name vlc instancia-original
```

  - **Comando:** Crear un snapshot llamando “vlc”, que es una copia de cómo era la imagen “instancia-original” en el momento de ejecutar el comando.

4.1.2.1.2. Creación de los descriptores

Para poder desplegar el Network Service a través de OSM son imprescindibles los descriptores VNFD y NSD. En este caso no se dispone de ellos y, por lo tanto, hay que crearlos desde cero.

- **VNFD:** Se muestra todo el contenido del fichero de configuración, seguido de una explicación de las sentencias más importantes:

vlc_vnfd.yaml

```
vnfd:vnfd-catalog:
    vnfd:
    -   id: vlc_vnfd
        name: vlc_vnf
        short-name: vlc_vnf
        description: VNF sobre ubuntu-cloud
        vendor: OSM
        version: '1.0'

        mgmt-interface:
            cp: eth0

        vdu:
        -   cloud-init-file: cloud_init.cfg
```

```
15              id: vlc_vnfd-VM
16              name: vlc_vnfd-VM
17              description: vlc_vnfd-VM
18              count: 1
19
20              vm-flavor:
21                 vcpu-count: 1
22                 memory-mb: 1536
23                 storage-gb: 10
24
25              image: vlc
26
27              interface:
28              -  name: eth0
29                 type: EXTERNAL
30                 virtual-interface:
31                    type: VIRTIO
32                    bandwidth: '0'
33                    vpci: 0000:00:0a.0
34                 external-connection-point-ref: eth0
35           connection-point:
36              -  name: eth0
37                 type: VPORT
```

Explicación de las sentencias más relevantes:

- **Sentencia 7 (“vendor”):** Indica la plataforma en la cual se va a usar el descriptor, OSM en este caso.
- **Sentencias 20-23 (“vm-flavor”):** Indica los requisitos hardware que hay proporcionarle a la nueva instancia:
  - **“vcpu-count”:** Indica el número de CPUs.
  - **“memory-mb”:** Indica la memoria RAM.
  - **“storage-gb”:** Indica el espacio de almacenamiento.
- **Sentencia 26 (“image”):** Nombre de la imagen que va a usar la instancia. En este caso es la imagen “vlc” creada en el apartado 4.1.2.1.1.
- **Sentencia 14 (“cloud-init file”):** Indica el nombre del fichero que contiene la configuración que tiene que aplicarse a la nueva instancia durante el arranque.

cloud-init.cfg

```
1  #cloud-config
2  password: osm.1234
3  chpasswd: { expire: False }
4  ssh_pwauth: True
```

Lo más relevante de este fichero es que se proporciona una clave al usuario por defecto, por lo tanto, ya no sería obligatorio acceder a la instancia por SSH y se podría acceder también a través de OpenStack.

- **NSD:** Se muestra todo el contenido del fichero de configuración, seguido de una explicación de las sentencias más importantes:

vlc_nsd.yaml

```
1  nsd:nsd-catalog:
2      nsd:
3      - id: vlc_nsd
4        name: vlc_ns
5        short-name: vlc_ns
6        description: Creado para TFG
7        vendor: OSM
8        version: '1.0'
9
10       constituent-vnfd:
11       - member-vnf-index: 1
12         vnfd-id-ref: vlc_vnfd
13
14       vld:
15         - id: vlc_nsd_vld1
16           name: vlc_nsd_vld1
17           short-name: vlc_nsd_vld1
18           type: ELAN
19           mgmt-network: 'true'
20           vim-network-name: provider
21           vnfd-connection-point-ref:
22           - member-vnf-index-ref: 1
23             vnfd-id-ref: vlc_vnfd
24             vnfd-connection-point-ref: eth0
```

Explicación de las sentencias más relevantes:

- **Sentencia 7 (“vendor”):** Indica la plataforma en la cual se va a usar el descriptor.
- **Sentencia 11 (“member-vnf-index**”): Indica el número de VNF que forma el Servicio de Red.
- **Sentencia 20 (“vim-network-name”):** Indica el nombre de la red virtual sobre la cual debe crearse la instancia.

4.1.2.2. Instanciación

Una vez creados los descriptores y la imagen se dispone de todos los elementos necesarios para crear la instancia.

- **Incorporación de los descriptores:** Para poder desplegar el NS hay que incluir los descriptores VNFD y NSD en OSM.

```
1 osm# osm upload-package vlc_vnf.tar.gz
2 osm# osm upload-package vlc_ns.tar.gz
```

  - **Comando1:** Incluir el nuevo descriptor en el catálogo de VNFDs disponibles de OSM.
  - **Comando 2:** Incluir el nuevo descriptor en el catálogo de NSDs disponibles de OSM.
- **Instanciación del NS:** Cuando estén todos los descriptores incorporados a OSM se puede crear el servicio de red.

```
1 osm# osm ns-create --nsd_name vlc_ns --ns_name emisor-video --vim_account openstack-site
2 osm# osm ns-create --nsd_name vlc_ns --ns_name recodificador-video --vim_account openstack-site
```

  - **Comando 1:** Crear un nuevo servicio de red llamado "emisor-video" usando como módulo VIM "openstack-site".
  - **Comando 2:** Crear un nuevo servicio de red llamado "recodificador-video" usando como módulo VIM "openstack-site".

Cabe destacar que tener que crear dos veces el NS para crear la misma instancia no es la solución más eficiente, ya que se podrían configurar los descriptores NSD y VNFD para que con una única creación del NS se pudieran obtener las dos VNFs necesarias, al igual que ocurría en el caso de las instancias CirrOS en el escenario de prueba. Sin embargo, el servidor de video necesita unos recursos más elevados y al intentar crear las VNFs se produce un fallo. Este problema no se trata de una limitación o un problema de OSM, sino del equipo sobre el que se está realizando el TFG que, con unos recursos bastante limitados, tiene que ejecutar la máquina virtual de OSM y las dos de los nodos controller y compute de OpenStack y además, se pretende que cree dos instancias "pesadas" al mismo tiempo. Por ese motivo se ha optado por ejecutar dos veces el mismo NS que, pese a no ser la solución más eficiente, es una alternativa efectiva.

4.1.2.3. Evaluación del Network Service

Las pruebas "estándar" ya se hicieron sobre el escenario de ejemplo, en el apartado 4.1.1.2., así que en este NS se pueden realizar directamente comprobaciones más complejas. En este caso se realiza una única prueba que consiste en evaluar la recodificación de video.

La emisión del video se realiza por un puerto concreto, así que la primera configuración que hay que hacer es modificar el grupo de seguridad "default" para permitir el tráfico entrante en el puerto en que se vaya a emitir. Cualquier puerto, de los no reservados, es una opción válida, en este caso se ha elegido el puerto 8080. También

hay que permitir el tráfico *HTTP*[24] porque es el protocolo por el que se realiza la transmisión.

```
1 controller#  openstack security group rule create --proto tcp --dst-port 8080 default
2 controller#  openstack security group rule create --proto tcp --dst-port 80 default
```

- **Comando 1:** Añadir al grupo de seguridad "default" una regla que permite el tráfico entrante por el puerto 8080.
- **Comando 2:** Añadir al grupo de seguridad "default" una regla que permite el tráfico por el puerto 80, el correspondiente al protocolo HTTP.

Realizada la configuración, la VNF destinada a la emisión del video puede comenzar a transmitir, según se indica en [39].

```
1 emisor-video#   vlc -vvv -I <nombre-video> --sout '#transcode{}:http{mux=asf,dest=:8080/}'
```

- **Comando:** Emitir el vídeo, sin ninguna modificación, por el puerto 8080.

Antes de recodificar el vídeo es conveniente visualizarlo sin ninguna modificación para verificar que la transmisión funciona correctamente. Para visualizar el vídeo se va a usar VLC, pero en este caso su versión gráfica, ya que se va a visualizar desde el equipo host. Para comenzar a ver el vídeo simplemente hay que introducir la URL, que contiene el protocolo de emisión, la dirección IP de la instancia que está emitiendo y el puerto seleccionado, es decir, http://<ip_emisor-video>:8080. En la Fig. 4.6 se muestra cómo se visualiza el vídeo original.

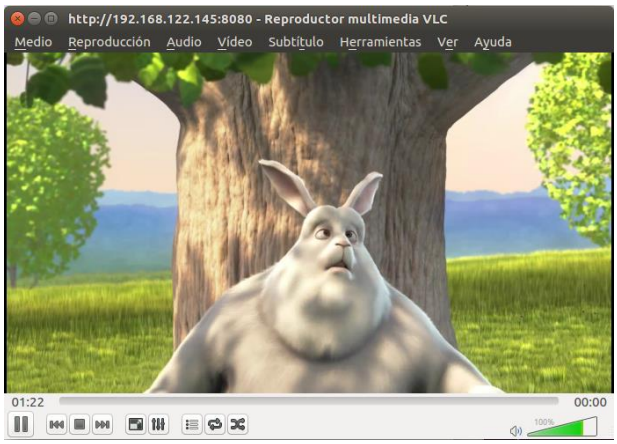

Fig. 4.6. Visualización vídeo original

Una vez se ha verificado que se transmite correctamente, la otra VNF puede comenzar la recodificación, que en este caso va a consistir en bajar la tasa a la que se transmite el vídeo. Aunque esta modificación en un primer momento pueda parecer inútil, ya que lo lógico es aumentarla para ver mejor el vídeo, sí que se trata de una opción interesante para algunos usuarios que cuenten con recursos limitados y sean incapaces de visualizarlo a su tasa original.

[24] HTTP, abreviación de Hypertext Transfer Protocol. Es el protocolo que permite la transferencia de hipertexto a través de internet.

```
1 recodifcador-video#  vlc -vvv -I http://<ip_emisor-video>:8080 --sout
                       '#transcode{vocdec=mp4v,vb=1512,fps=30,scale=Automático}:http{mux=avi,
                       dest=:8080/}'
```

- **Comando:** Modificar la tasa del vídeo transmitido por la otra VNF y volver a emitirlo por el puerto 8080.

Para visualizar el video recodificado hay que introducir nuevamente la URL en VLC, pero en este caso con la IP de la instancia que se encarga de disminuir la tasa del vídeo: http://<ip_recodificador-video>:8080. En la Fig. 4.7 se muestra cómo se visualiza el vídeo recodificado, cuya calidad ha disminuido respecto al original.

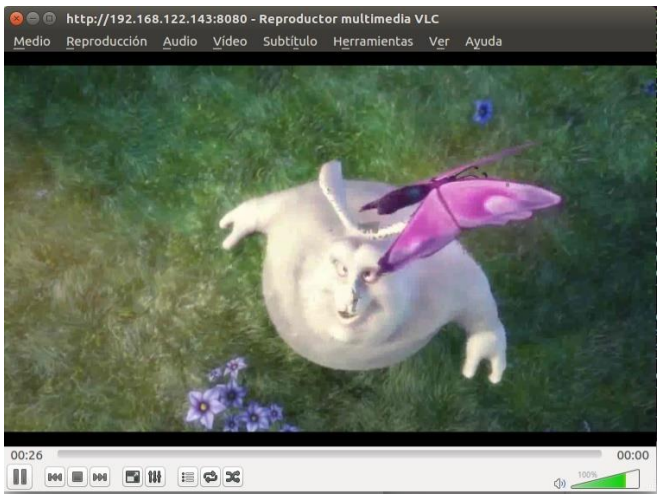

Fig. 4.7. Visualización del video recodificado

De esta manera queda completado este NS, permitiendo a los usuarios con buena conexión a internet ver el vídeo a su tasa original y a los que tengan una conexión peor, ver el vídeo a una tasa más asequible.

4.1.3. Network Service 3: Recodificación de vídeo avanzada

El NS anterior tiene un inconveniente y es que los usuarios solo pueden visualizar el vídeo a la vez que se está emitiendo, al igual que ocurre con la televisión. Una mejora interesante es incorporar una VNF que contenga un servidor de vídeo bajo demanda, así los usuarios pueden solicitar la visualización de un vídeo en cualquier momento, en vez de poder verlo cuando se esté transmitiendo, sería un comportamiento similar al de la plataforma YouTube. La otra VNF hace la recodificación de una manera similar solo que en esta ocasión, en vez de cambiar la tasa del vídeo se cambia la forma de emisión, pasando de transmitir en unicast a multicast.

El procedimiento para llegar a la solución es muy similar al utilizado en el escenario anterior. No se van a repetir los pasos que sean muy similares, por lo que se centrará este apartado en los cambios realmente relevantes como son la obtención de la imagen y las pruebas realizadas al NS.

4.1.3.1. Diseño

De nuevo, el apartado de diseño tiene una gran relevancia para este NS, ya que no se dispone de los elementos necesarios para poder desplegarlo y, por lo tanto, hay que crearlos.

4.1.3.1.1. Creación de la imagen

Para este NS también hay que obtener una imagen con el software ya instalado, así se puede instanciar directamente sin tener que repetir configuración. Para el NS anterior era solo recomendable, sin embargo, en este caso es prácticamente obligatorio, ya que el procedimiento de instalación del servidor de vídeo bajo demanda es mucho más complejo y largo que el de VLC.

Para implementar el servidor de vídeo bajo demanda se ha elegido Ampache, el cual es una alternativa de software libre que proporciona este servicio. El procedimiento para instalarlo [40], partiendo de la imagen original de Ubuntu-Cloud, es el siguiente:

- **Instalación de servidor web *Apache*[25]:** Ampache funciona sobre un servidor web, que hay que instalar.

```
instancia#  apt-get install apache2
instancia#  service apache2 status
instancia#  apt-get install php7.0 libapache2-mod-php7.0 php7.0-xml php7.0-curl php7.0-gd
instancia#  php -v
```

  - **Comando 1:** Instalar Apache, opción elegida para implementar el servidor web.
  - **Comando 2:** Comprobar que el software se ha instalado y funciona correctamente.
  - **Comando 3:** Instalar las librerías *PHP*[26] requeridas para que el servidor web pueda alojar a Ampache.
  - **Comando 4:** Comprobar la correcta instalación de PHP.
- **Instalación MySQL:** El servidor streaming bajo demanda necesita una base de datos para almacenar los videos.

```
instancia#  apt-get install mysql-server-5.7
instancia#  service mysql status
instancia#  Apt-get install php7.0-mysql
```

  - **Comando 1:** Instala MySQL, software elegido para implementar la base de datos.
  - **Comando 2:** Comprobación de que se ha instalado correctamente.
  - **Comando 3:** Integración de PHP con MySQL.

[25] Es el servidor web más utilizado. Además, es de código abierto y uso gratuito.
[26] Es un lenguaje de programación utilizado para desarrollo web.

- **Descargar Ampache:** Descarga del software principal del servidor de video bajo demanda.

```
1 instancia# wget https://github.com/ampache/ampache/archive/3.8.4.tar.gz
2 instancia# tar -xf 3.8.4.tar.gz
3 instancia# mv ampache-3.8.4 /var/www/html
```

  - **Comando 1:** Descargar la última versión de Ampache.
  - **Comando 2:** Descomprimir los archivos recién descargados.
  - **Comando 3:** Mover los ficheros de Ampache para que la aplicación sea accesible desde servidor web.
- **Instalación del manejador de dependencias Composer:** Es una herramienta necesaria a la hora de instalar Ampache.

```
1 instancia# wget https://getcomposer.org/installer
2 instancia# php installer
3 instancia# Mv composer.phar /usr/local/bin/composer
4 instancia# Composer ---version
```

  - **Comando 1:** Descargar Composer.
  - **Comando 2:** Instalar Composer.
  - **Comando 3:** Mover el manejador de dependencias a una ubicación desde la cual se pueda tener acceso global.
- **Instalación de Ampache**: Instalación del servidor de video mediante la herramienta Composer.

```
1 instancia# chmod 777 -R /var/www/html/ampache-3.8.4/
2 instancia# cd /var/www/html/ampche-3.8.4
3 instancia# composer install –-prefer-source –-no-interaction
```

  - **Comando 1:** Otorgar permisos de ejecución al directorio de Ampache.
  - **Comando 3:** Instalación de Ampache y sus dependencias.

Concluida la instalación de Ampache se puede obtener la imagen modificada haciendo un snapshot.

```
1 controller# openstack snapshot create –name ampache instancia-original
```

- **Comando:** Crear un snapshot llamando “ampache”, que es una copia de cómo era la imagen “instancia-original” en el momento de ejecutar el comando.

4.1.3.1.2. Creación de los descriptores

Los descriptores VNFD y NSD de este nuevo escenario son prácticamente iguales a los del NS anterior (apartado 4.1.2.1.2.), así que incluirlos de nuevo sería repetir información de manera innecesaria. Lo único que hay que tener en cuenta es que ahora no se está usando la imagen “vlc”, así que se debe modificar la sentencia 26 del fichero “vnfd.yaml” para que la imagen sea “ampache”. También es recomendable cambiar cada

alusión a “vlc” por la palabra “ampache”, simplemente por ser consecuente con la nomenclatura, no afecta a la ejecución de la instancia.

4.1.3.2. Instanciación

Una vez creados tanto los descriptores como la imagen, se disponen de todos los elementos necesarios para crear la instancia.

- **Incorporación de los descriptores:** Para poder desplegar el NS hay que incluir los descriptores VNFD y NSD en OSM.

```
1 osm# osm upload-package ampache_vnf.tar.gz
2 osm# osm upload-package ampache_ns.tar.gz
```

  - **Comando1:** Incluir el nuevo descriptor en el catálogo de VNFDs disponibles de OSM.
  - **Comando 2:** Incluir el nuevo descriptor en el catálogo de NSDs disponibles de OSM.
- **Instanciación del NS:** Cuando ya estén todos los descriptores incorporados a OSM se puede crear el servicio de red.

```
1 osm# osm ns-create --nsd_name ampache_ns --ns_name servidor-video --vim_account
    openstack-site
2 osm# osm ns-create --nsd_name vlc_ns --ns_name recodificador-video --vim_account
    openstack-site
```

  - **Comando 1:** Crear un nuevo servicio de red llamado “servidor-video” usando como módulo VIM “openstack-site”.
  - **Comando 2:** Crear un nuevo servicio de red llamado “recodificador-video” usando como módulo VIM “openstack-site”, es el NS creado en el escenario anterior que hace las funciones de recodificación.

4.1.3.3. Evaluación del Network Service

Las pruebas que se realizan a este NS consisten en constatar que las dos VNFs funcionan, tanto la que contiene el servidor de video bajo demanda como la encargada de la recodificación.

Primero se debe comprobar que el servidor streaming funciona correctamente, para ello basta con acceder a la plataforma y suministrarle un vídeo que pueda incorporar a su catálogo para así tener algo que compartir. Para acceder a Ampache hay que introducir la siguiente URL en un navegador web: http://<ip_instancia>/ampache-3.8.4. En la Fig. 4.8 se muestra la página de inicio de sesión en la plataforma.

Fig. 4.8. Inicio de sesión en Ampache

Dentro de la plataforma ya se puede incorporar un vídeo de ejemplo a su catálogo, que debe hacerse a través de la interfaz gráfica, por lo que el procedimiento [41] se mostrará a partir de capturas de pantalla.

- **Añadir un catálogo:** En la parte izquierda de la interfaz hay un menú lateral que contiene la opción de "Añadir un catálogo". Está situada en la parte superior de dicho menú como se muestra en la Fig. 4.9.

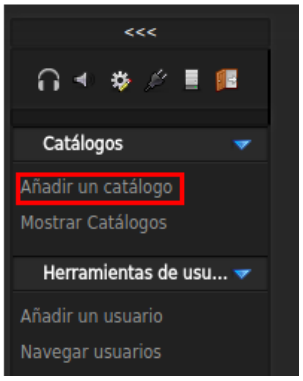

Fig. 4.9. Opción "Añadir un catálogo"

- **Incluir vídeos:** Seguidamente aparece el formulario mostrado en la Fig. 4.10, en el cual hay que rellenar los siguientes campos:
  - **"Nombre del catálogo":** Es obligatorio proporcionar un nombre.
  - **"Tipo de catálogo":** Hay que elegir la opción "local", ya que el vídeo se encuentra en la VNF.
  - **"Ruta":** Es la ubicación en la que se encuentran los vídeos que se quieren incluir en el catálogo, en este caso solo uno.

Fig. 4.10. Incluir un vídeo al servidor Ampache

Con un vídeo en el catálogo, el servidor ya tiene algo que poder suministrar a los usuarios. Para probar que la plataforma proporciona correctamente el vídeo bajo demanda

se puede visionar desde la propia GUI de Ampache, como se muestra en la Fig. 4.11, o desde VLC, como se hizo en el apartado 4.1.2.3.

Fig. 4.11. Visualización del vídeo desde Ampache

Comprobada la correcta transmisión del vídeo se puede probar la recodificación, función que hace la segunda VNF. Esto se hace mediante el siguiente comando:

```
recodificador-video# vlc -vvv -I dummy "http://192.168.122.147/ampache-3.8.4/play/index.php?ssid=981bbeaf825b4012eac2e6c5fe2a5bd7&type=video&oid=2&uid=1&name=Big%20Buck%20Bunny%20%2030mb.mp4" --sout '#transcode{}:rtp{dst=224.13.13.13,port=5004,mux=ts}"
```

- **Comando:** Recodifica el video almacenado en el servidor Ampache y lo emite por el grupo multicast 224.13.13.13.

Para ver el vídeo recodificado hay que introducir la siguiente URL en VLC: rtp://224.13.13.13:5004, como se muestra en la figura 4.12.

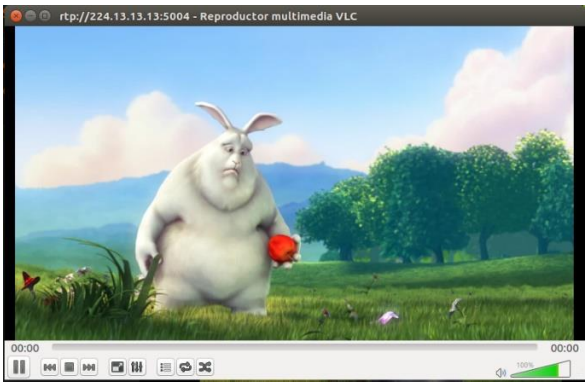

Fig. 4.12. Visualización de vídeo multicast

4.2. Evaluación de la plataforma OSM

En este apartado se realiza la evaluación de Open Source MANO, la cual está centrada en un análisis del rendimiento de la plataforma. Para realizar una evaluación más exhaustiva, también se prueban otras funcionalidades poco comunes que incorpora OSM y lo que diferencian del resto de orquestadores, estas funcionalidades son la inclusión de aplicaciones complementarias en la propia instalación de la plataforma y la funcionalidad VIM-emulator.

4.2.1. Análisis de rendimiento

Para determinar el rendimiento de la plataforma se han evaluado los tiempos de despliegue de:

- La propia plataforma.
- NS sencillos.
- NS complejos.

A la hora de medir los tiempos se realizan múltiples mediciones para calcular el valor medio, que es el más adecuado. El cálculo del valor medio se realiza con la *Ecuación* 4.1.

$$\bar{x} = \frac{1}{N}\sum_{i}^{N} x_i \tag{4.1}$$

Se sigue el siguiente criterio [42] para saber el número de medidas necesarias que hay que realizar:

1. Se realizan tres medidas y se calcula su valor medio $\bar{x}$.
2. Se calcula el porcentaje de dispersión según la *Ecuación* 4.2.

$$D = \frac{|x_{max} - x_{min}|}{\bar{x}} \times 100 \tag{4.2}$$

3. Dependiendo del valor obtenido tras calcular el porcentaje de dispersión media hay que realizar el siguiente número de medidas:
    - Si D < 2%, con las tres medidas realizadas es suficiente.
    - Si 2% ≤ D < 8%, hay que realizar un total de seis medidas.
    - Si 8% ≤ D ≤ 15 %, hay que realizar un total de 15 medidas.
    - Si D > 15%, hay que realizar un total de 50 medidas.

Es importante destacar que los tres aspectos elegidos para determinar el rendimiento de la plataforma no son los únicos que podrían haberse evaluado, ya que existen varios que podrían analizarse para determinar el rendimiento de OSM. Incluso dentro de los tres aspectos elegidos existe una gran variabilidad debido a:

- Existen varias partes del proceso que pueden elegirse a la hora de medir los tiempos.
- Existen varios procedimientos que permiten medir los tiempos.
- Las condiciones de la implementación, ya que no es lo mismo un entorno de pruebas con un equipo con recursos limitados, como el usado para este estudio, que en un entorno de producción que cuenta con servidores con muchos más recursos.

Por todo ello, la solución obtenida en este estudio no es la única existente y, por lo tanto, las conclusiones que se extraigan tampoco son las únicas.

4.2.1.1. Tiempo de despliegue de OSM

Hay que tener en cuenta que las medidas realizadas en este apartado se han hecho en las condiciones más favorables posibles, es decir, todos los recursos del equipo host están libres y disponibles para la creación de la VM de OSM. Si por ejemplo el equipo estuviera ejecutando las VMs de OpenStack, los tiempos obtenidos serían algo superiores.

Cada medida es el tiempo trascurrido entre:

- **Inicio:** El arranque de la máquina virtual de OSM, es decir, el instante en que se presiona el botón de arranque de la VM en el interfaz gráfico de KVM.
- **Fin:** La plataforma OSM ya está operativa. Aunque la VM esté encendida y se pueda interactuar con ella, los servicios de OSM tardan unos segundos más en estar disponibles. Por ello se ha elegido el instante en que el acceso a Launchpad, la GUI de OSM, está disponible para comprobar que OSM ya está totalmente operativo.

En la Tabla 4.1 se recogen las tres primeras medidas, las cuales permiten determinar el número adecuado de mediciones que se deben realizar.

Número de medida (i)	**Tiempo de despliegue**	**Tiempo de despliegue (s) (x_i)**
1	0’ 42’’ 44	42,44
2	0’ 41’’ 18	41,18 (x_{min})
3	0’ 44’’ 03	44,03 (x_{max})

Tabla 4.1. Tres primeras medidas del tiempo de despliegue de OSM

Aplicando la *Ecuación* 4.2 con los valores de la Tabla 4.1, se obtiene que D=6,70% y, por lo tanto, son necesarias seis medidas para obtener un valor medio adecuado. En la Tabla 4.2 se muestran las seis medidas, incluyendo las tres primeras ya realizadas.

Número de medida (i)	**Tiempo de despliegue**	**Tiempo de despliegue (s) (x_i)**
1	0’ 42’’ 44	42,44
2	0’ 41’’ 18	41,18
3	0’ 44’’ 03	44,03
4	0’ 44’’ 78	44,78
5	0’ 42’’ 43	42,43
6	0’ 43’’ 26	43,26

Tabla 4.2. Medidas para calcular tiempo de despliegue de OSM

Aplicando la *Ecuación* 4.1., se obtiene que el valor medio es 43,02 s, que es un tiempo de despliegue rápido si lo comparamos con el obtenido en la versión anterior de OSM. Con dicha versión se realizó una única medida de 1 min 15,74 s que, aunque no tan fiable como los valores obtenidos por el procedimiento anterior, sí permite hacerse

una idea de la mejora realizada por la plataforma, reduciendo casi a la mitad su tiempo de despliegue.

4.2.1.2. Tiempo de instanciación de Network Services sencillos

En el apartado 4.2.1.1. se tomaron los tiempos en las condiciones más favorables que se podían conseguir en el equipo host, sin embargo, a la hora de calcular los tiempos de instanciación de NS no es posible, ya que las tres VMs tienen que estar en funcionamiento, una para OSM y dos para OpenStack.

Se ha elegido el NS proporcionado por OSM, evaluado en el apartado 4.1.1., como ejemplo de NS sencillo a la hora de medir los tiempos de instanciación. Cada medida es el tiempo transcurrido desde:

- **Inicio:** OSM solicita la creación del NS, es decir, el instante en el que se presiona el botón de creación del NS en el interfaz gráfico de OSM.
- **Fin:** Las VNFs correspondientes al NS están totalmente creadas, es decir, el instante en que la instancia solicita al usuario que se identifique.

En la Tabla 4.3 se recogen las tres primeras medidas, las cuales permiten determinar el número adecuado de mediciones que se deben realizar.

Número de medida (i)	**Tiempo de despliegue (min)**	**Tiempo de despliegue (s)** (x_i)	
1	0’ 42’’ 14	42,14	(x_{max})
2	0’ 39’’ 62	39,62	
3	0’ 37’’ 58	37,58	(x_{min})

Tabla 4.3. Tres primeras medidas del tiempo de despliegue de un NS sencillo

Aplicando la $Ecuación$ 4.2 con los valores de la Tabla 4.3, se obtiene que D=11,46% y, por lo tanto, son necesarias 15 medidas para obtener un valor medio apropiado. En la Tabla 4.4 se muestran las 15 medidas, incluyendo las 3 primeras ya realizadas.

Número de medida (i)	Tiempo de despliegue	Tiempo de despliegue (s) (x_i)
1	0’ 42’’ 14	42,14
2	0’ 39’’ 62	39,62
3	0’ 37’’ 58	37,58
4	0’ 38’’ 60	38,60
5	0’ 36’’ 71	36,71
6	0’41’’ 19	41,19
7	0’44’’ 45	44,45
8	0’ 41’’ 74	41,74
9	0’ 39’’ 16	39,16
10	0’ 38’’ 46	38,46
11	0’ 40’’ 94	40,94
12	0’ 42’’ 35	42,35
13	0’ 37’’ 81	37,81
14	0’ 41’’ 67	41,67
15	0’ 40’’ 25	40,25

Tabla 4.4. Medidas para calcular tiempo de instanciación de un NS sencillo

Aplicando la *Ecuación* 4.1, se obtiene que el valor medio es 40,18 s. Aunque se trate de un tiempo razonable hay que tener en cuenta que la medida obtenida es meramente cualitativa, ya que, en un servidor con unos recursos superiores se obtendrían unos tiempos de despliegue inferiores.

4.2.1.3. Tiempo de instanciación de Network Services complejos

Se ha elegido el NS de recodificación de vídeo avanzada, evaluado en el apartado 4.1.3., como ejemplo de NS complejo a la hora de medir los tiempos de instanciación. Cada medida es el tiempo trascurrido desde:

- **Inicio:** OSM solicita la creación del NS, es decir, el instante en el cual se presiona el botón de creación del NS en el interfaz gráfico de OSM.
- **Fin:** Las VNFs correspondientes al NS están totalmente creadas, es decir, el instante en que la instancia solicita al usuario que se identifique.

En la Tabla 4.5 se recogen las tres primeras medidas, las cuales permiten determinar el número total de ellas que son necesarias para obtener un valor medio adecuado.

Número de medida (i)	Tiempo de despliegue (min)	Tiempo de despliegue (s) (x_i)
1	2’ 36’’ 10	156,10 (x_{max})
2	2’ 31’’ 27	151,27
3	2’ 25’’ 42	145,42 (x_{min})

Tabla 4.5. Tres primeras medidas del tiempo de despliegue de un NS complejo

Aplicando la *Ecuación* 4.2, con los valores de la Tabla 4.5, se obtiene que D=7,08% y, por lo tanto, son necesarias seis medidas para obtener un valor medio apropiado. En la Tabla 4.6 se muestran las seis medidas, incluyendo las 3 primeras ya realizadas.

Número de medida (i)	Tiempo de despliegue	Tiempo de despliegue (s) (x_i)
1	2’ 36’’ 10	156,10
2	2’ 31’’ 27	151,27
3	2’ 25’’ 42	145,42
4	2’ 24’’ 03	144,03
5	2’ 23’’ 36	143,36
6	2’ 25’’ 89	145,89

Tabla 4.6. Medidas para calcular tiempo de instanciación de un NS complejo

Aplicando la *Ecuación* 4.1, se obtiene que el valor medio es 147,68 s, o lo que es lo mismo 2 min 27,68 s. En este caso el valor medio obtenido es bastante superior comparado con los casos anteriores, superando los dos minutos de duración, no obstante, hay que seguir teniendo presente que la medida obtenida es meramente cualitativa, ya que, en un servidor con unos recursos superiores se obtendrían unos tiempos de despliegue inferiores.

4.2.2. Aplicaciones complementarias

La integración con aplicaciones externas a través de la propia instalación de OSM es una de las características que diferencia a esta plataforma del resto de orquestadores. Estas dos aplicaciones complementarias son Grafana y Kibana.

4.2.2.1. Grafana

Existen dos formas de instalar Grafana en OSM [43]:

- **Instalarla a la vez que OSM.**

```
1 osm#    ./install_osm.sh --pm_stack
```

  - **Comando:** Instalar Grafana junto a OSM.

- **Instalarla sobre una implementación de OSM versión 4 existente.**

```
1 osm#    ./install_osm.sh -o pm_stack
```

- **Comando: Instalar** Grafana sobre OSM.

Como es lógico, en este caso se ha seleccionado instalar Grafana sobre la implementación de OSM existente, ya que el resultado obtenido por los dos procedimientos es exactamente el mismo y, por lo tanto, no tiene sentido repetir la instalación de la plataforma.

Para acceder a Grafana, Fig. 4.13, hay que introducir en un navegador web la dirección IP de la VM que aloja a OSM, procedimiento comentado en el apartado 3.3.4., que se utilizó para acceder a Launchpad. No obstante, en este caso se precisa del puerto 3000, para distinguir el acceso a Launchpad del acceso a Grafana. Por lo tanto, hay que introducir la siguiente dirección en un navegador web: http://192.168.122.21:3000.

Fig. 4.13: Página de acceso a Grafana

En la *Fig. 4.14*[27] se muestra un ejemplo de lo que se puede hacer con Grafana, en este caso ver el uso de CPU y memoria de las instancias.

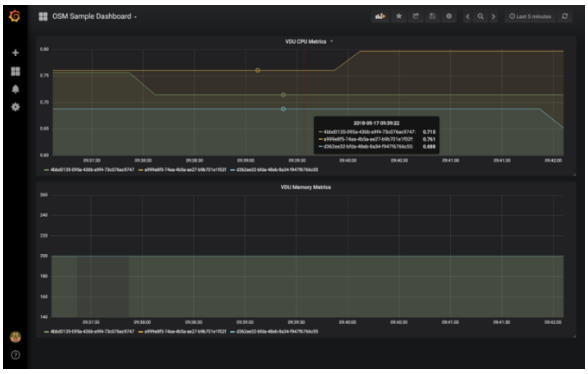

Fig. 4.14: Ejemplo de visualización de métricas en Grafana [44]

4.2.2.2. Kibana

Al igual que ocurría con Grafana, existen dos formas de instalar Kibana [45] en OSM:

- **Instalarla a la vez que OSM.**

```
1 osm#   ./install_osm.sh --elk_stack
```

  - **Comando:** Instalar Kibana junto a OSM.

[27] La figura no ha sido obtenida experimentalmente, sino que ha sido extraída de la página web de OSM. Lo apropiado hubiera sido poder evaluar Grafana, sin embargo, por problemas con el módulo MON de OSM no ha sido posible.

- **Instalarla sobre una implementación de OSM versión 4 existente.**

```
osm#  ./install_osm.sh -o elk_stack
```

  - **Comando:** Instalar Kibana sobre OSM.

Para acceder a Kibana hay que introducir nuevamente la dirección IP de la VM que aloja a OSM en un navegador web, solo que en este caso el puerto de Kibana es el 5601: http://192.168.122.21:5601.

En la *Fig. 4.15*[28], obtenida de la página web de OSM, se muestra un ejemplo de uso de la plataforma, en este caso la visualización de alarmas surgidas por el alto uso de CPU por parte de una instancia.

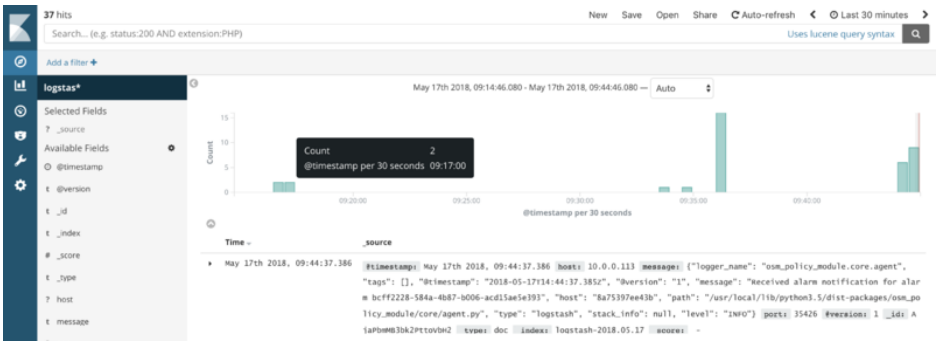

Fig. 4.15. Ejemplo de uso de Kibana [46]

4.2.3. VIM-emulator

En el apartado 2.3.1.7. se comentó que existían cuatro tecnologías diferentes que recomendaba OSM para implementar el módulo VIM. No obstante, desde la versión FOUR de la plataforma, se ofrece una posibilidad llamada VIM-emulator [47], la cual permite no tener que asociar un VIM junto a la arquitectura de OSM ya que se emula dicho módulo en la propia plataforma. Se trata de una función muy útil ya que permite probar rápidamente una arquitectura MANO completa, sin necesidad de crear un VIM, que es un proceso complejo como se puede comprobar en el Anexo D.

El procedimiento [48] para instalar VIM-emulator es el siguiente:

- **Instalación:**

```
osm#  ./install_osm.sh -o vimemu
```

  - **Comando:** Instalar VIM-emulator junto a OSM.
- **Asociar el VIM a OSM:**

```
osm#  export VIMEMU_HOSTNAME=$(sudo docker inspect -f '{{range
      .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' vim-emu)
osm#  osm vim-create --name emu-vim1 --user username --password password --auth_url
      http://$VIMEMU_HOSTNAME:6001/v2.0 --tenant tenantName --account_type
      openstack
```

[28] La figura no ha sido obtenida experimentalmente, sino que ha sido extraída de la página web de OSM. Lo apropiado hubiera sido poder evaluar Kibana, sin embargo, por problemas con el módulo MON de OSM no ha sido posible.

- **Comando 1:** Crear la variable de entorno “VIMEMU_HOSTNAME”, la cual se necesita para conocer la dirección IP del VIM-emultor.
- **Comando 2:** Asociar el nuevo módulo VIM a OSM.

Una vez finalizado el procedimiento anterior, VIM-emulator queda instalado y, por lo tanto, se cuenta con una arquitectura MANO completa, sin embargo, para evaluar que funciona correctamente es necesario instanciar VNFs en la nueva arquitectura. Para ello se ha utilizado un NS especialmente diseñado para VIM-emulator, el cual ha sido proporcionado por OSM y que aún no ha sido mencionado en este estudio.

El procedimiento para utilizar el NS es el siguiente:

- **Incluir los descriptores NSD y VNFD:**

```
1 osm#  git clone https://osm.etsi.org/gerrit/osm/vim-emu.git
2 osm#  osm vnfd-create vim-emu/examples/vnfs/ping.tar.gz
3 osm#  osm vnfd-create vim-emu/examples/vnfs/pong.tar.gz
4 osm#  osm nsd-create vim-emu/examples/services/pingpong_nsd.tar.gz
```

  - **Comando 1:** Descargar los descriptores.
  - **Comando 2 y 3**: Añadir los VNFDs a OSM.
  - **Comando 4**: Añadir el NSD a OSM.
- **Crear el NS:**

```
1 osm#  osm ns-create --nsd_name pingpong --ns_name test --vim_account emu-vim1
```

  - **Comando:** Crear el NS correspondiente a VIM-emulator.

Una vez que se ha creado el NS queda demostrado que la funcionalidad VIM-emulator permite obtener una arquitectura MANO, sin un VIM dedicado y que las funciones de este módulo pueden estar emuladas dentro del propio orquestador.

Capítulo 5

Planificación y análisis del entorno socio-económico

En este capítulo se muestra la planificación realizada para poder llevar a cabo el proyecto. También se incluye un análisis del entorno socio-económico, que contiene el presupuesto de elaboración del TFG y un análisis del impacto socio-económico esperado de la aplicación del mismo.

5.1. Planificación

En este apartado se detalla la planificación de las tareas realizadas para este proyecto. En la Tabla 5.1 se detalla el tiempo previsto para cada bloque, que son agrupaciones de tareas con una temática similar. Para el cálculo se ha estimado que cada mes cuenta con 20 días laborables y se va a trabajar, de media, lo mismo que un trabajador a media jornada, es decir 4 horas diarias, por lo que un mes equivale a 80 horas de trabajo.

Tarea	Tiempo (Horas)	Duración (Semanas)
A. Documentación teórica inicial	**30**	**1,5**
A.1. NFV.	15	
A.2.OSM.	10	
A.3. Módulo VIM.	5	
B. Máquinas virtuales	**10**	**0,5**
B.1. Documentación sobre entornos de virtualización.	3	
B.2. Encontrar el entorno de virtualización apropiado para el proyecto.	5	
B.3. Instalación del entorno y aprendizaje de creación de VMs.	2	
C. OSMv3	**50**	**2,5**
C.1. Documentación sobre la instalación del software.	10	
C.2. Instalación del software.	35	
C.3. Toma de contacto con la plataforma.	5	
D. Módulo VIM	**70**	**3,5**
D.1. Documentación sobre la instalación del software.	10	
D.2. Encontrar el módulo VIM más apropiado para OSM.	55	
D.3. Toma de contacto con la plataforma.	5	
E. Integración OSMv3 y módulo VIM	**20**	**1**
E.1. Configuración VIM para ser compatible con OSMv3.	10	
E.2. Integración con OSMv3.	10	
F. Network Services (Inicio)	**20**	**1**
F.1. Aprendizaje de creación de NS en OSM.	20	
G. OSMv4	**30**	**2,5**
G.1. Documentación sobre la instalación del software.	3	
G.2. Instalación del software.	25	
G.4. Toma de contacto con la plataforma.	2	

H. Integración OSMv4 y OpenStack	**10**	**0,5**
H.1. Configurar OpenStack para ser compatible con OSMv4.	8	
H.2. Integración.	2	
I. Network Services (continuación)	**100**	**5**
I.1. Diseño de los NS.	40	
I.2. Creación de los NS.	30	
I.3. Evaluación de los NS.	30	
J. Evaluación	**50**	**2,5**
J.1. Análisis de rendimiento.	20	
J.2. Análisis de funcionalidades diferenciales	30	
K. Memoria	**160**	**8**
L. Presentación	**40**	**2**
	590	**30,5**

Tabla 5.1. Realización de tareas

Eso implica que la duración del proyecto ha sido algo más de 7 meses y que el número de horas trabajadas fue bastante superior a las horas requeridas para elaborar un TFG, calculadas con la $Ecuación\ 5.1$. Esto se debe a que una vez iniciado el proyecto fue necesario reelaborar la planificación inicial debido a que apareció una nueva actualización de la plataforma OSM.

$$Mínimo\ horas = 12\ créditos\ ECTS \times \frac{25\ horas}{1\ crédito\ ECTS} = 300\ horas \qquad (5.1)$$

A principios de mayo de 2018 salió a la luz la nueva versión de OSM y era importante evaluar las nuevas funcionalidades que incorporaba. Se estimó que instalar la versión FOUR de OSM iba a retrasar la finalización del proyecto tres semanas, por lo que era necesario decidir entre las siguientes opciones:

- No probar OSM versión FOUR y presentar el TFG en la convocatoria de julio.
- Probar OSM versión FOUR y estudiar solo lo que fuera posible de la plataforma antes de la convocatoria de julio.
- Probar OSM versión FOUR estudiando a fondo la plataforma y presentar el TFG en la convocatoria de octubre.

Finalmente se optó por instalar la nueva versión porque el objetivo principal del TFG consiste en realizar un estudio en profundidad de OSM y por lo tanto era imprescindible evaluarla. Por el mismo motivo se descartó evaluar la plataforma superficialmente.

Como consecuencia no se pudo finalizar el estudio antes de la convocatoria de julio y se podía dedicar más tiempo a estudiar la plataforma, creando NS más complejos y estudiando aspectos que diferenciaban a OSM del resto de orquestadores. En la Fig. 5.1 se muestra el diagrama de Gantt de la planificación del proyecto (en rojo se resalta el día en el que hubo que reestructurar la planificación).

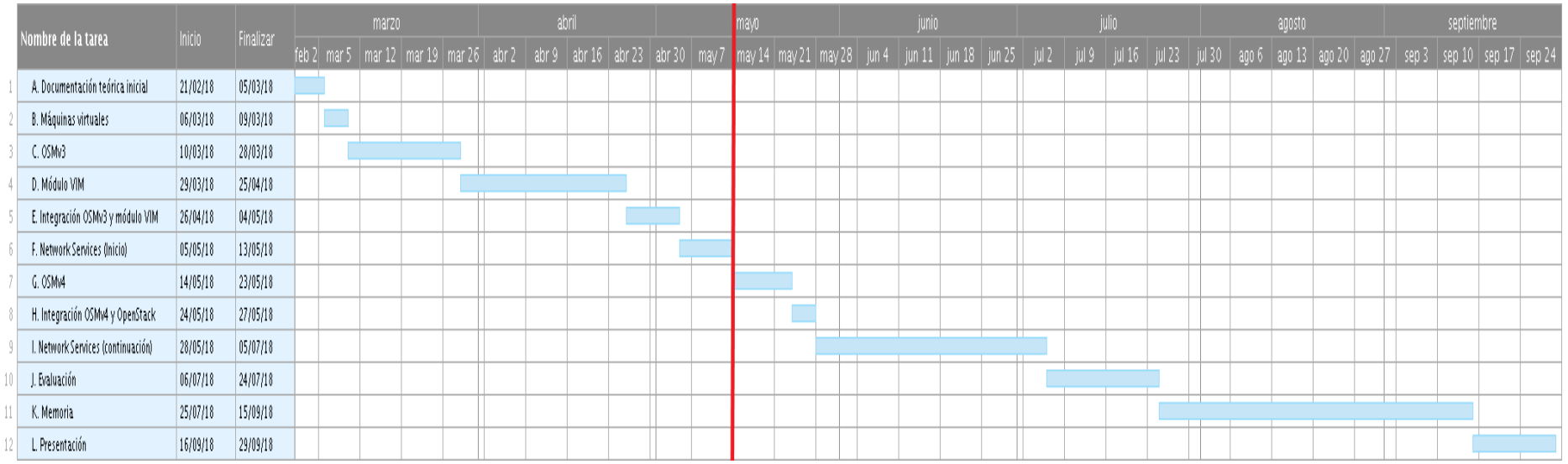

Fig. 5.1. Diagrama de Gantt de la planificación

5.2. Entorno socio-económico

Este apartado incluye el presupuesto de elaboración del TFG y un análisis del impacto socio-económico esperado de la aplicación del mismo.

5.2.1. Presupuesto del TFG

En esta sección se describen los costes del proyecto divididos en costes de recursos materiales y costes de recursos humanos.

5.2.1.1. Costes de recursos materiales

Los costes relacionados con recursos materiales se muestran desglosados en tres bloques: hardware, software y suministros.

Recursos hardware

Aparte de la utilización un portátil para realizar la solución técnica se ha usado un portátil adicional para hacer la memoria, el cual no era imprescindible. Al ser muy tedioso tener que intercalar en un único portátil dos sistemas operativos, uno que contenía la solución técnica y otro con la herramienta necesaria para realizar la memoria, se optó por usar dos portátiles para trabajar más eficientemente.

- **Ordenador portátil UltranoteIV:** Utilizado para el desarrollo de la solución técnica.
  - **Procesador:** Intel Core i7-7700HQ.
  - **Tarjeta gráfica:** Intel HD Graphics 620.
  - **Memoria RAM:** 16 GB DDR4.
  - **Almacenamiento:** SSD 480 GB.
- **Ordenador portátil Asus UX430UA:** Utilizado para la realización de la memoria.
  - **Procesador:** Intel Core i7-7500U.
  - **Tarjeta gráfica:** Intel HD Graphics 620.
  - **Memoria RAM:** 8 GB DDR4.
  - **Almacenamiento:** SSD 512 GB.

Recursos software

Se ha usado todo el software libre y gratuito posible para no gastar más de lo necesario en este aspecto.

- **Sistemas operativos:**
  - Windows 10 Home original de 64 bits.
  - Ubuntu 16.04 Desktop.
  - Ubuntu 16.04 Server.
  - Ubuntu 16.04 Cloud.
  - CirrOS.

- **Plataformas:**
  - Open Source MANO.
  - OpenStack.
- **Herramientas Web:**
  - Dropbox.
  - *Smartsheet*[29].
  - *Cacoo*[30].
- **Herramientas de Ofimática:**
  - *Microsoft Office 365*[31].

Suministros:

También hay que tener en cuenta otros gastos indirectos resultantes de la elaboración del proyecto.

- **Electricidad.**
- **Agua.**
- **Gas.**
- **Internet.**

En la Tabla 5.2 se muestran desglosados los gastos asociados a todos los recursos materiales.

Concepto	Cantidad	Coste (€)	Tiempo de uso (meses)	Vida útil (meses)	Dedicación[32] (%)	Total (€)
Portátil Ultranote	1	991	7	60	100	115,62
Portátil Asus	1	968	2	60	50	16,14
Windows 10	1	91	2	60	50	1,52
Electricidad + Agua + Gas[33]	1	50	7	-	30	105
Internet	1	20	7	-	30	42
						280,28

Tabla 5.2. Costes de recursos materiales

El cálculo de la columna "Total" ha sido realizado con la *Ecuación* 5.2.

[29] Herramienta web usada para la realización del diagrama de Gantt.
[30] Herramienta web usada para la realización de algunas figuras.
[31] Herramienta de pago que se ha considerado gratuita en el proyecto, debido a que ha sido proporcionada por la universidad.
[32] Esta columna indica cual ha sido el porcentaje de uso de un recurso durante el proyecto. Por ejemplo, no supone el mismo gasto un equipo exclusivamente dedicado al TFG que uno que también usado para hacer cosas ajenas al proyecto.
[33] Para calcular este gasto se ha tenido en cuenta lo que se considera gasto deducible fiscalmente para un autónomo que trabaja en su residencia que, aunque no es el mismo caso que el del autor del proyecto, es bastante similar. Corresponden al 30% de los gastos de suministro de agua, gas y electricidad proporcionales al área de trabajo de la vivienda del autónomo.

$$Total = \frac{Tiempo\ de\ uso}{Vida\ útil} \times Coste \times Dedicación \quad (5.2)$$

5.2.1.2. Costes de recursos humanos

Para los costes relacionados con recursos humanos se tienen en cuenta las horas trabajadas por el autor del proyecto, un *ingeniero junior*[34] y, por otro lado, el tutor y el director del TFG, dos ingenieros seniors.

Según el estudio "Perfil Socio Profesional del Ingeniero de Telecomunicaciones" [49] realizado por el Colegio Oficial de Ingenieros de Telecomunicaciones y la Asociación Española de Ingenieros de Telecomunicaciones, el salario medio de un ingeniero de telecomunicaciones es de 51.220 € anuales y para ingenieros con menos de 5 años de experiencia es de 23.553 € anuales.

En la Tabla 5.3 se muestran los gastos asociados a los recursos humanos.

Concepto	Cantidad	Tiempo (horas)	*Coste hora*[35] (€)	Total (€)
Ingeniero junior	1	590	13,09	7.720,15
Ingeniero senior	2	*29,5*[36]	28,46	1.679,14
				9.399,29

Tabla 5.3. Costes de recursos humanos

5.2.1.3. Costes totales

El presupuesto total del proyecto se indica en la Tabla 5.4:

Concepto	Total (€)
Costes de recursos materiales	280,28
Costes de recursos humanos	9.399,29
	9.679,57

Tabla 5.4. Costes totales

5.3. Impacto socio-económico

Este TFG trata del estudio teórico de una plataforma de orquestación NFV, en este caso OSM. Una posible aplicación práctica del trabajo podría ser la siguiente: una operadora, está planeando utilizar en su red la tecnología NFV y necesita un orquestador, el cual es parte importante de la arquitectura que tiene que desplegar. Existen varias alternativas en el mercado para implementarlo, por lo que la operadora necesita evaluarlas para ver cuál es la que mejor se adapta a sus necesidades.

[34] Ingeniero que tiene menos de 5 años de experiencia en el sector.
[35] Coste calculado al dividir el sueldo mensual entre la estimación de las horas trabajadas al año, 1800.
[36] Corresponde al 5% de las horas trabajadas por el autor del proyecto.

Por lo tanto, este TFG donde mejor encajaría es en el área de investigación y, como puede pasar con otros proyectos de este departamento, cabe la posibilidad de que lo que se investiga no suponga un beneficio económico para la empresa. En el escenario hipotético planteado, se evaluaban varios orquestadores, de los cuales es probable que el operador finalmente acabe implementando uno o dos de ellos, y eso supone que todo lo invertido en el estudio de los otros orquestadores, aunque sea necesario hacerlos, no aporte beneficios. En el peor de los casos, este estudio puede suponer un impacto negativo de 9.679,57 € para la empresa si finalmente se opta por utilizar otro orquestador, sin embargo, hay que tener en cuenta que OSM se postula como una de las mejores alternativas del sector y, lo que sí está claro, es que invertir en la tecnología NFV es una apuesta segura.

NFV ofrece grandes oportunidades para los operadores de red, proveedores de soluciones y, lo que es más importante, en la apertura de nuevos modelos de negocio y oportunidades de innovación.

Como tecnología emergente, NFV plantea varios retos como por ejemplo: la garantía del rendimiento de la red para los dispositivos virtuales, su instanciación, migración dinámica y colocación eficiente. Además, supone inversión de tiempo, recursos y educación, sin embargo, puede ser la estrategia más eficiente para posicionarse en el sector, el cual se enfrenta a una competencia cada vez mayor a medida que las nuevas empresas se incorporan a las nuevas tecnologías.

A continuación, se enumeran algunas de las ventajas económicas [1] más importantes que se prevén obtener al usar NFV. Están agrupadas dependiendo de si suponen una reducción de la inversión requerida (CAPEX) o una reducción del coste de operación (OPEX):

- **Reducción del CAPEX:**
  - Se reducen los costes en equipamiento, al necesitar menos equipos especializados que son sustituidos por servidores genéricos.
  - Los proveedores de servicios de red pueden compartir la infraestructura.
- **Reducción del OPEX:**
  - Mayor velocidad de despliegue de nuevos servicios al minimizar el ciclo de innovación de los operadores de red, lo cual permite rentabilizar antes la inversión.
  - Los servicios pueden ampliarse o reducirse rápidamente según sea necesario, es decir, se invierten los recursos en la demanda real.
  - Aprovisionamiento remoto en software sin necesidad de realizar visitas al sitio físico para instalar nuevo hardware, las cuales son costosas.
  - Reducción del consumo de energía, ya que sería posible concentrar la carga de trabajo en un número menor de servidores durante las horas de poco uso. De este modo todos los demás servidores pueden apagarse o ponerse en modo de ahorro de energía.

En resumen, NFV aporta rentabilidad, mejoras en el tiempo de comercialización, nuevos modelos de negocio y una mayor innovación en la infraestructura y aplicaciones de la industria de las telecomunicaciones. Todas esas ventajas se verán traducidas en reducciones del gasto de capital y del gasto de explotación que oscilan entre el 3.7% y el 9% del coste total [2] que, aunque son cifras significativas, parecen ser algo inferiores a las expectativas de la industria.

Capítulo 6

Conclusión y trabajos futuros

En este capítulo se exponen las conclusiones obtenidas de la realización de este Trabajo Fin de Grado. También se proponen ideas para trabajos futuros sobre OSM.

6.1. Conclusión

Para un ingeniero junior, sin experiencia en el sector, el nivel de dificultad de este proyecto es elevado, debido principalmente a la gran cantidad de conocimientos requeridos en áreas muy diferentes. Hace falta tener, o adquirir en este caso, conocimientos sobre creación y uso de máquinas virtuales, comandos de Ubuntu, tecnología NFV y la plataforma OpenStack.

No obstante, a pesar de la dificultad de algunas tareas, todos los objetivos del proyecto han sido completados, pero lo más importante, es que ha permitido afianzar y adquirir conocimientos, que es la verdadera finalidad del TFG.

A continuación, se exponen las conclusiones que se han extraído del estudio de OSM, haciendo especial énfasis en las virtudes y defectos de la plataforma que han sido detectados durante su uso.

La utilización de dos versiones distintas durante este estudio ha permitido comprobar las mejoras de la plataforma, principalmente en los siguientes aspectos:

- **Recursos hardware:** Entre las dos últimas versiones se ha producido una reducción muy significativa en el consumo de recursos hardware, que se ha visto reducidos a la mitad en la mayoría de los casos. Estos cambios no estaban tan enfocados para entornos de producción, sino más bien a entornos de pruebas con recursos limitados, como es el caso de este estudio. Se agradece el esfuerzo que ha hecho la plataforma para mejorar este aspecto, ya que ha permitido investigarla con más profundidad.
- **Instalación:** La instalación de la plataforma se ha simplificado mucho respecto a versiones anteriores, reduciendo casi al mínimo la configuración previa necesaria para poder comenzar la instalación. Esto supone que la plataforma sea más accesible y pueda ser implementada por más usuarios y operadores de red.

Sin embargo, todavía hay algunos aspectos de la plataforma que se pueden mejorar:

- **Perdida de funcionalidades:** OSM FOUR ha mejorado en muchos aspectos respecto a la versión anterior, sin embargo, hay uno de ellos que era mejor antes. En la versión THREE se incluía una herramienta llamada “VNF/NS Catalog Composer”, la cual permitía completar y crear los descriptores NSD y VNFD mediante el uso de unos formularios. Solo se podía acceder a ella a través de Launchpad y permitía desarrollar rápidamente y con precisión los descriptores de la entidad que se estuviera modelando. Se trataba de una funcionalidad muy interesante que ha desaparecido en la versión FOUR, por lo que ahora hay que hacer los descriptores de manera manual, que es mucho más complejo ya que hay tener un mayor conocimiento de ellos.
- **Documentación:** La documentación proporcionada por OSM, es en general muy completa y clara, algo que es de agradecer. Sin embargo, se ha echado en falta documentación complementaria para poder usar más fácilmente MON, el módulo de monitorización de la plataforma. Para poder usarlo hay que configurar el VIM

elegido para soportar dicha monitorización e incluso instalar otras aplicaciones externas. Es entendible que no se proporcione dicha documentación, ya que el módulo VIM no es algo que pertenezca directamente a la arquitectura de OSM. No obstante, dado que el grupo de trabajo de esta plataforma se ha esforzado tanto en mejorar este módulo, sería una pena que no pudiese usarse por falta de documentación, por lo que incluirla, o al menos indicar dónde encontrarla, parece una mejora interesante de cara al futuro.

En cualquier caso, OSM cuenta con muchas más virtudes que defectos y pretende seguir mejorando, sacando para ello, nuevas versiones al mercado cada pocos meses.

Pese a ser una plataforma relativamente joven, con únicamente dos años de vida, se postula como una de las alternativas más importantes en la orquestación de servicios en infraestructuras NFV y no cabe duda de que los operadores la tendrán muy en cuenta cuando estén preparados para el lanzamiento de las primeras funciones virtualizadas.

6.2. Trabajos futuros

La experiencia adquirida durante el estudio de OSM permite proponer nuevas líneas de estudio para posibles proyectos futuros de la plataforma:

- **Virtual Infraestructure Manager:** OSM ofrece la posibilidad de elegir el módulo VIM entre varias opciones. Para este proyecto solo se ha estudiado la implementación de la plataforma con OpenStack, por lo que podría ser interesante integrar la plataforma con otras tecnologías VIM diferentes.
- **Módulo MON:** Utilizar el módulo de monitorización de la arquitectura de OSM permite diseñar escenarios muy interesantes, como por ejemplo crear instancias de manera dinámica si aumenta la demanda por parte de los usuarios. Durante este estudio no se ha podido investigar suficientemente dicho módulo por problemas a la hora de instalar complementos en OpenStack, pero es probable que esos problemas se solucionen en el futuro y se pueda estudiar MON más a fondo en posteriores estudios.

También se incluyen una serie de consejos que pueden resultar útiles si se planea continuar con el estudio de la plataforma:

- **Hardware potente:** Aunque cada vez OSM necesite menos recursos hardware para su puesta en funcionamiento, se recomienda contar con un equipo de especificaciones elevadas, ya que permitirá implementar escenarios más complejos y así poder estudiar más a fondo la plataforma.
- **Actualizaciones:** OSM es una plataforma que se actualiza cada pocos meses, por lo que es probable que una actualización coincida durante la realización del estudio. Aunque en un primer momento pueda parecer un contratiempo al dejar obsoleta gran parte de la información usada, puede ser en realidad una ventaja, ya que es probable que alguna dificultad o problema encontrado sea resuelto por la nueva versión. Por ello se recomienda invertir algo de tiempo en investigar la

actualización y trabajar sobre ella, que no solo permitirá realizar un estudio más actualizado, sino también más completo.

Capítulo 7

Anexos

Uno de los objetivos de este proyecto es hacer una memoria autocontenida, es decir, que en ella aparezca documentado todo lo necesario para poder replicar el estudio realizado. La finalidad de este capítulo es recoger todos los procedimientos que, aunque importantes para llegar a la solución, no tenían cabida en el Capítulo 3, debido a su larga extensión o no estar lo suficientemente relacionados con el tema principal de este proyecto.

7.1. Anexo A: Procedimiento de instalación de KVM

El procedimiento [50] de instalación de la herramienta es el siguiente: en primer lugar, hay que asegurarse de que el equipo sobre el cual se quiere instalar KVM soporta la virtualización, para ello hace falta ejecutar los siguientes comandos:

```
1 host#  apt install cpu-checker
2 host$  kvm-ok
```

- **Comando 1:** Instalar el paquete "cpu-checker", que permite determinar si el equipo usado es apto para la virtualización.
- **Comando 2:** Evaluar si el equipo soporta la virtualización y por lo tanto se puede instalar KVM. La salida deseada debe ser similar a la mostrada en la Fig. 7.1.

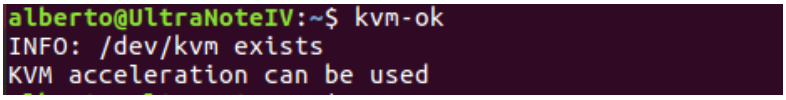

Fig. 7.1. Comprobación de la instalación de KVM

Si el equipo es apto para la virtualización se puede proceder con la instalación de KVM.

```
1 host#  apt-get install qemu-kvm libvirt-bin virt-manager
```

- **Comando:** Instalar los KVM y los paquetes complementarios necesarios.

7.2. Anexo B: Creación de una máquina virtual en KVM

Para la creación de una máquina virtual en la herramienta de virtualización KVM se ha optado por mostrar el procedimiento [50] a través del interfaz gráfico de la aplicación. En cada etapa de creación se muestra una captura de pantalla, con el fin de ayudar a comprender en que menú del wizard hay que realizar las acciones que se indican a través del texto.

El procedimiento para crear una máquina virtual en KVM es el siguiente:

- **Fig. 7.2:** Elegir la opción de crear a partir de Imagen ISO.

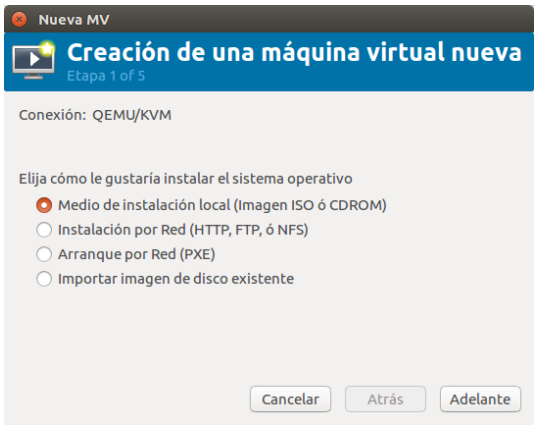

Fig. 7.2. Creación VM – Imagen OS

- **Fig. 7.3:** Seleccionar la imagen ISO del sistema operativo que va a tener la nueva máquina virtual, en este caso de Ubuntu 16.04 server.

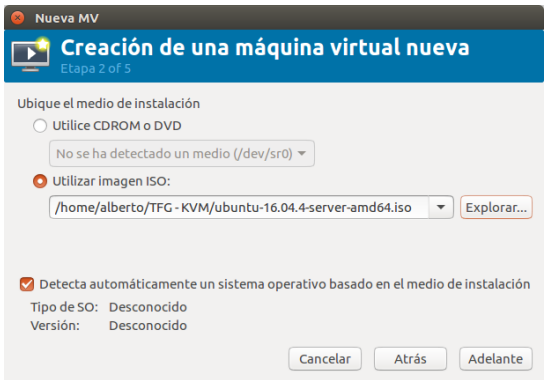

Fig. 7.3. Creación VM – Elección OS

- **Fig. 7.4:** Seleccionar la memoria RAM y el número de CPUs que se quiere proporcionar a la nueva VM.

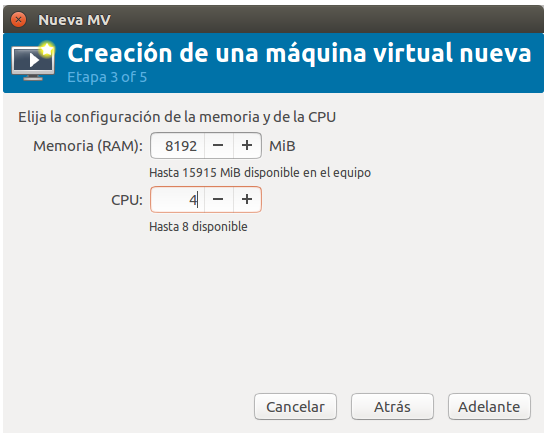

Fig. 7.4. Creación VM – Elección RAM y CPU

- **Fig. 7.5:** Seleccionar la cantidad de almacenamiento que se quiere asignar a la máquina virtual.

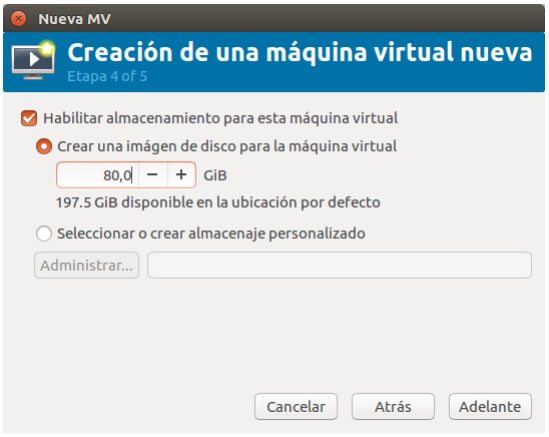

Fig. 7.5. Creación VM – Elección almacenamiento

- **Fig. 7.6:** Elegir el nombre de la máquina virtual. También se puede cambiar el tipo de interfaz de red, aunque se recomienda dejar el valor por defecto para todas las VM que se necesitan crear en este proyecto.

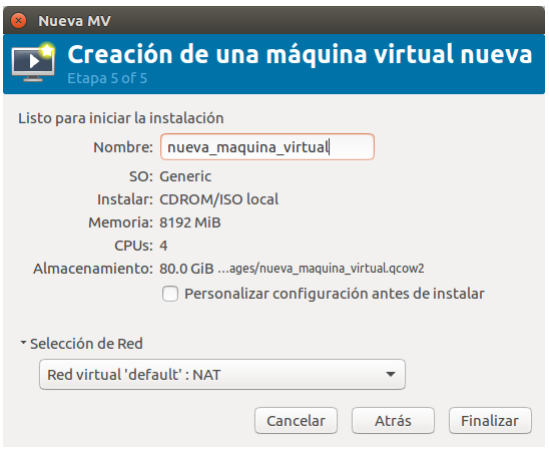

Fig. 7.6. Creación VM – Elección nombre

7.3. Anexo C: Instalación del sistema operativo Ubuntu 16.04 Server

La instalación del sistema operativo Ubuntu-Server hay que realizarla obligatoriamente por interfaz gráfico por lo que este anexo indica, mediante capturas de pantalla, que datos hay que introducir en cada etapa del wizard. Esto implica que la información estará principalmente contenida en las figuras, que normalmente no estarán acompañadas de texto explicativo, debido a que preguntan cosas muy sencillas como pueden ser el idioma, el nombre del usuario, etc. No obstante, sí se explican los apartados más complejos, cuya respuesta no es tan trivial.

El procedimiento de instalación del sistema operativo es el siguiente:

- **Selección del idioma del wizard de instalación.**

Fig. 7.7. Selección del idioma instalación

- **Comienzo de la instalación del sistema operativo.**

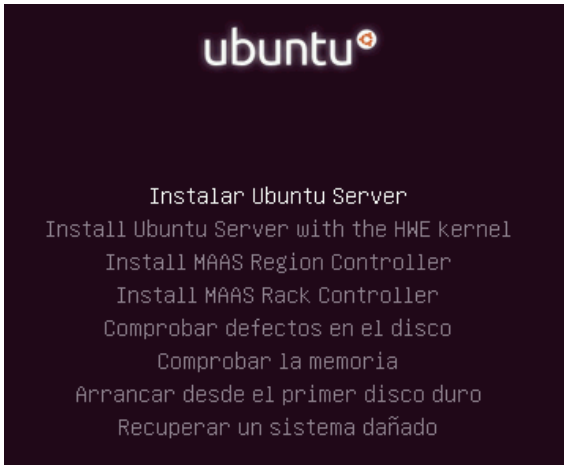

Fig. 7.8. Comenzar la instalación

- **Elección del país de ubicación.**

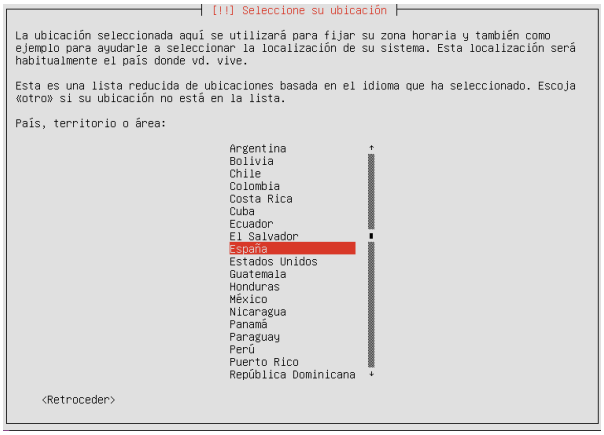

Fig. 7.9. Elección de la ubicación

- **Configuración del idioma del teclado.**

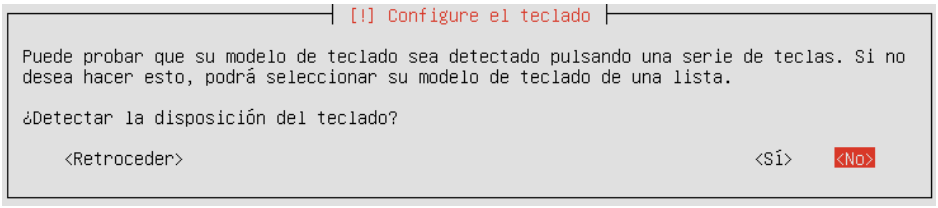

Fig. 7.10. Configuración del teclado – Paso 1

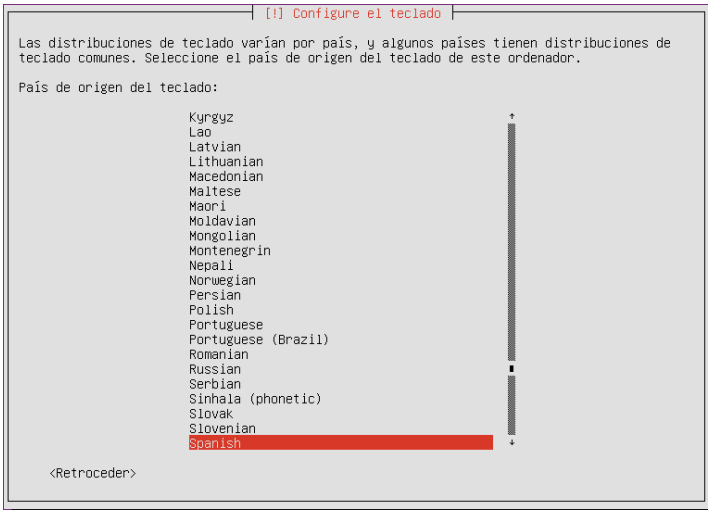

Fig. 7.11. Configuración del teclado – Paso 2

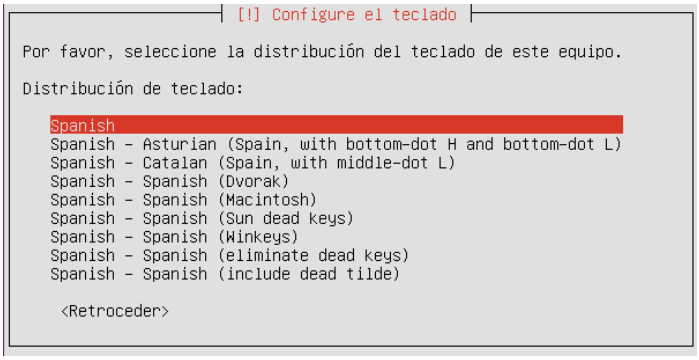

Fig. 7.12. Configuración del teclado – Paso 3

- **Elección del nombre de la máquina.**

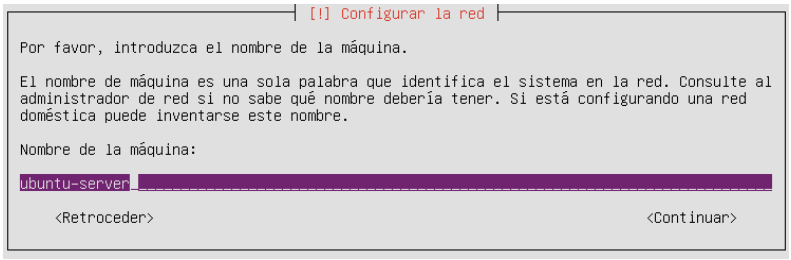

Fig. 7.13. Elección del nombre del equipo

- **Configuración de usuario y contraseña.**

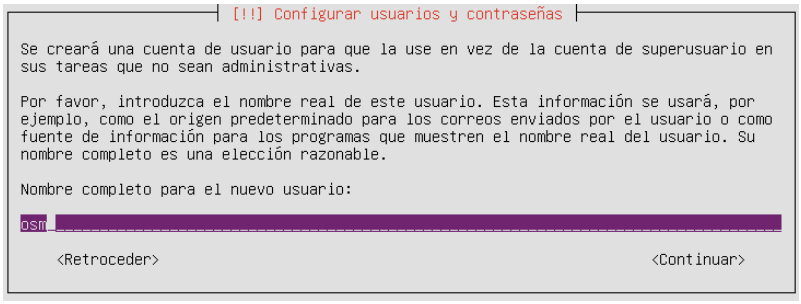

Fig. 7.14. Elección del nombre del superusuario

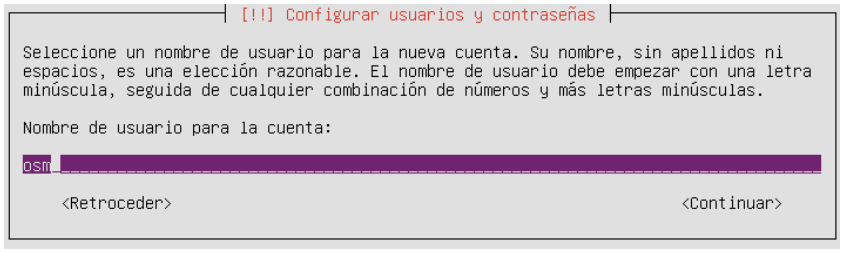

Fig. 7.15. Elección del nombre de la cuenta de usuario

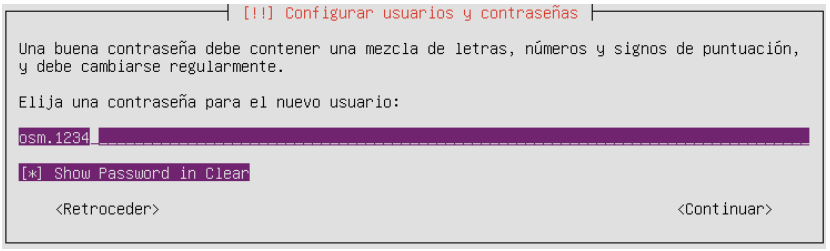

Fig. 7.16. Configuración de la contraseña del usuario

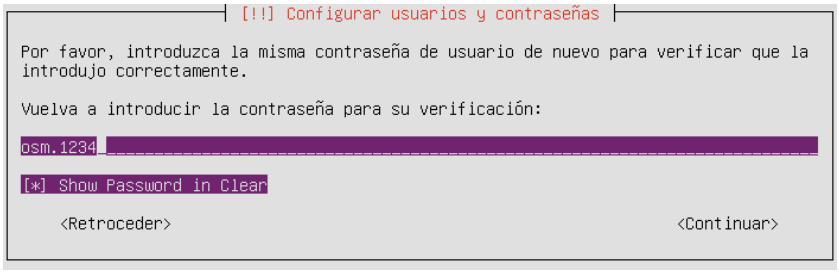

Fig. 7.17. Confirmación de la contraseña

- **Cifrado de carpeta personal:** Seleccionar esta opción implica un pequeño descenso en el rendimiento del sistema, ya que se cifra y descifra cada archivo que esté relacionado con esa cuenta, pero a cambio, en caso de extraviar el ordenador, será más complicado robar los datos que contiene. A pesar de ser una opción muy recomendable, al tratarse de una máquina virtual y necesitar que tenga el máximo rendimiento posible, la mejor elección es no cifrar.

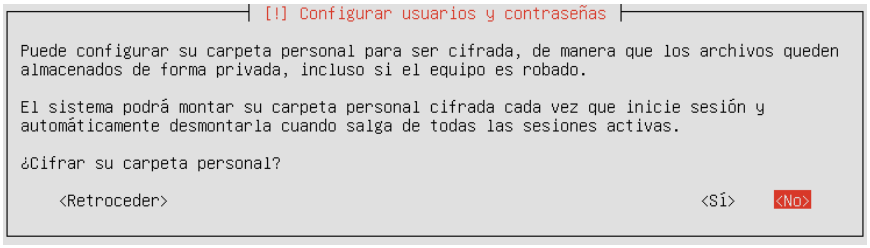

Fig. 7.18. Cifrado de carpeta personal

- **Configuración de reloj.**

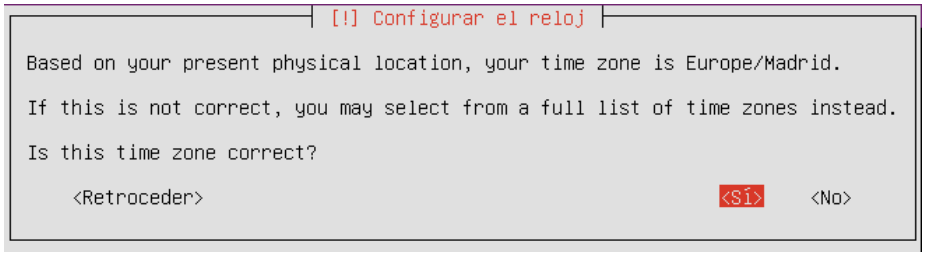

Fig. 7.19. Configuración del reloj

- **Particionado de discos:** Se trata de una de las partes más críticas de la instalación, ya que hacerlo mal podría implicar falta de memoria en algunas de las particiones e incluso que no funcione el servidor. Como no se precisa de una distribución concreta, no hace falta configurar el particionado de manera manual. Configurar LVM (Logical Volume Manager) [51] permite en el futuro redimensionar las particiones en caso de que alguna se quede pequeña, por ello es la opción más recomendable. Cifrar tiene ventajas de seguridad, pero disminuye el rendimiento, por lo que se descarta esa opción. Por todo ello la opción más adecuada es "Guiado - utilizar el disco completo y configurar LVM."

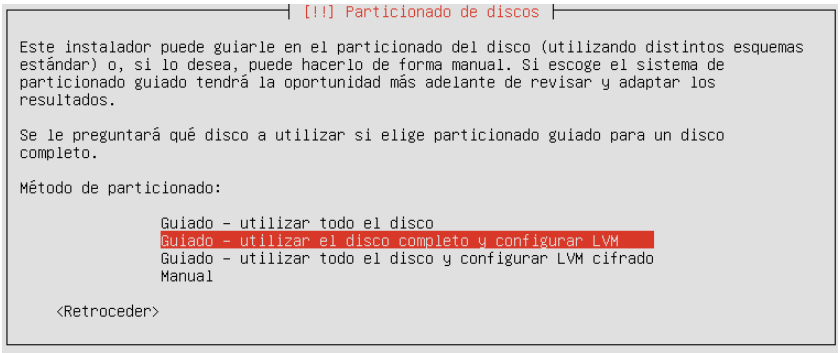

Fig. 7.20. Elección del método de particionado

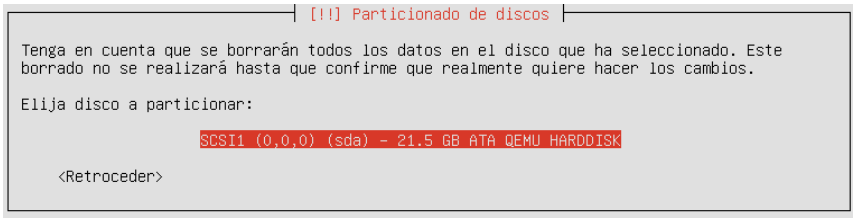

Fig. 7.21. Elección del disco a particionar – Paso 1

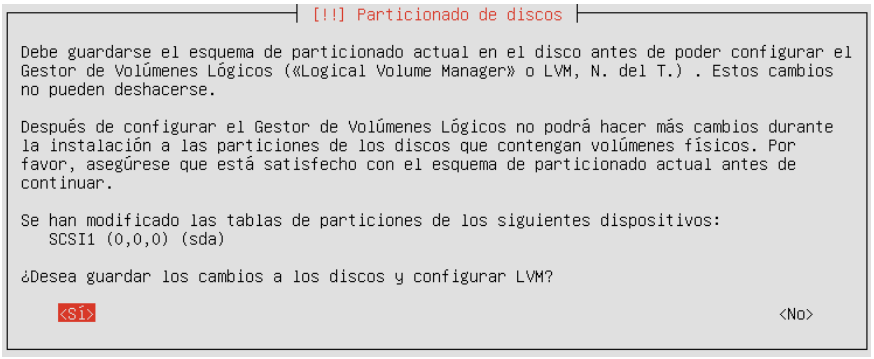

Fig. 7.22. Elección del disco a particionar – Paso 2

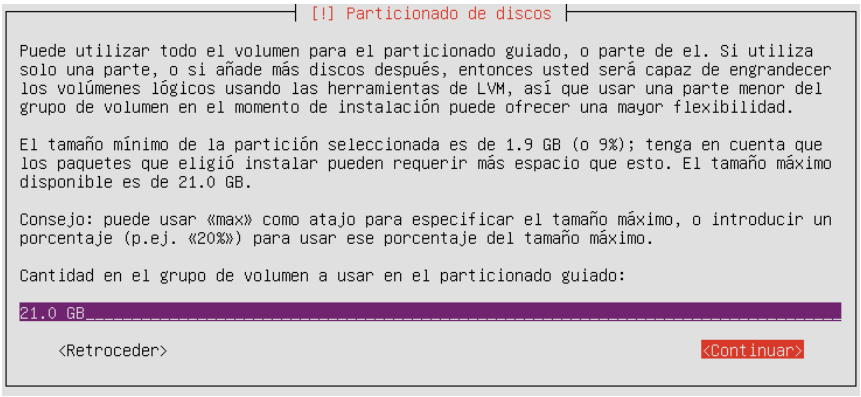

Fig. 7.23. Selección del tamaño a particionar

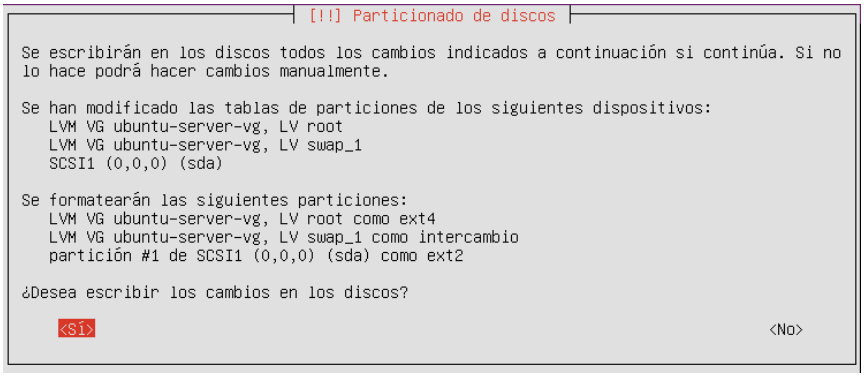

Fig. 7.24. Confirmación de particionado de disco

- **Configuración de proxy HTTP:** Lo normal es no usar un proxy por lo que hay que dejar sin responder, en caso contrario habría que indicarlo con el formato especificado.

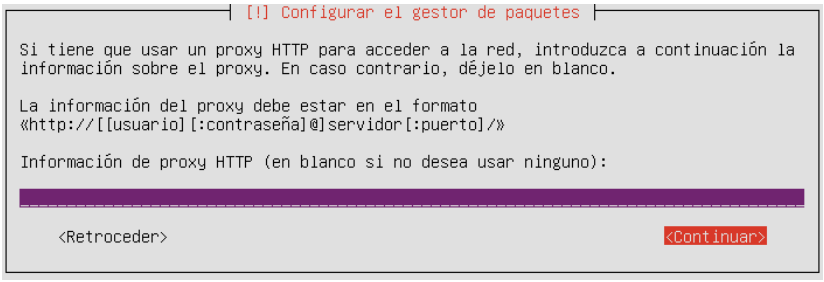

Fig. 7.25. Confirmación de proxy HTTP

- **Actualizaciones del equipo:** Lo más recomendable es hacer las actualizaciones manualmente, así en caso de que alguna actualización afecte a algún componente del proyecto y este deje de funcionar, será más sencillo identificarla y encontrar una solución.

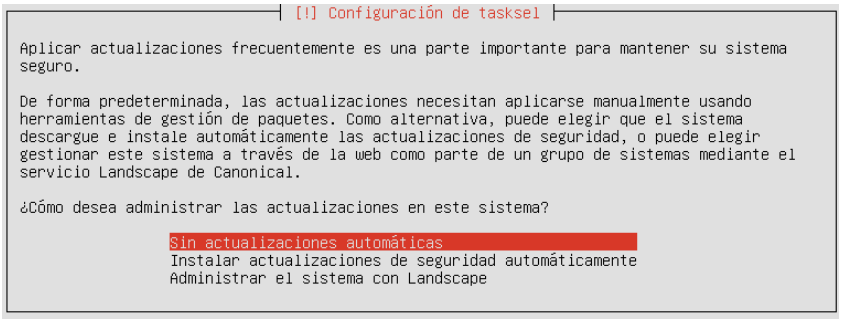

Fig. 7.26. Elección de cómo se debe actualizar el equipo

- **Instalación de programas adicionales:** Aparte de la instalación de los elementos básicos, el wizard también ofrece la posibilidad de instalar algunos de los paquetes más comunes en este tipo de servidores. Una opción muy útil es "OpenSSH server" ya que permitirá conectarse por SSH a la máquina virtual, y así poder trabajar con ella más fácilmente. En cualquier caso, podremos instalar cualquier programa adicional después.

Fig. 7.27. Instalación de programas adicionales

- **Instalación del cargador de arranque:** GRUB (GRand Unifier Bootloader) [52] es un gestor de arranque que permite tener en un mismo equipo diferentes sistemas operativos.

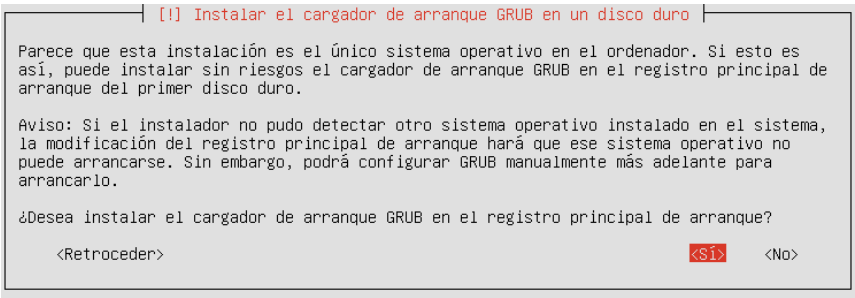

Fig. 7.28. Instalación de GRUB

- **Finalizar la instalación.**

Fig. 7.29. Finalizar la instalación

Una vez concluido el proceso se reinicia es sistema y se puede comenzar a usar la máquina virtual

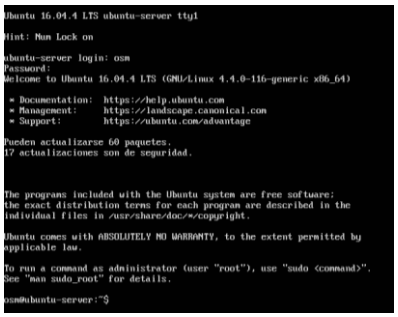

Fig. 7.30. Primer uso de la máquina virtual

7.4. Anexo D: Procedimiento de instalación de OpenStack

Este anexo es un procedimiento [53] que, aunque importante para llegar a la solución, es complementario a la tecnología principal del proyecto. Por lo tanto, los pasos no se explican tan detalladamente como los de otros capítulos, donde cada línea de código era analizada.

En cada apartado se indica de manera explícita el nodo de la arquitectura, controller o compute, sobre el cual hay que aplicar la configuración, con el fin de facilitar el seguimiento del proceso.

7.4.1. Configuración del entorno

En esta sección se explica la configuración básica que hay que aplicar a los nodos controller y compute, la cual es la base de los servicios más complejos de OpenStack.

7.4.1.1. Seguridad

Configuración del nodo controller

OpenStack proporciona varios servicios y la mayoría de ellos necesitan tener asociada una contraseña para conseguir una mayor seguridad en la implementación. Para generar claves seguras, hay que ejecutar el siguiente comando (una vez por cada servicio):

```
1 controller#  openssl rand -hex 10
```

Una vez generadas todas las claves, 15 en total, es recomendable guardarlas en un documento de texto y así poder usarlas más adelante.

```
1 controller#  vim /root/passwordlist
```

/root/passwordlist

```
1  export DB_PASS=ae6e757af896e81fb26
2  export ADMIN_PASS=9cbcde54016694167ac9
3  export CINDER_DBPASS=ab067969aa7e39dbad21
4  export CINDER_PASS=10111fa0540ce7a63eba
5  export DASH_DBPASS=86919efebc129ff0e70f
6  export DEMO_PASS=11d649a6714d3380626a
7  export GLANCE_DBPASS=cda9bd2eecea3d47a670
8  export GLANCE_PASS=c10218b2e490f5822f6d
9  export KEYSTONE_DBPASS=c1d3b017c5ceb579b8e2
10 export NEUTRON_DBPASS=98317a93b286dbc55259
11 export NEUTRON_PASS=a23423041cf257196070
12 export NOVA_DBPASS=d95dc71b40111b6fcf16
13 export NOVA_PASS=8676dc35c60fc95b10a3
14 export RABBIT_PASS=0733cadb7ace0d5e7a40
15 export METADATA_SECRET=0abb3f64fa1034271455
```

La manera más útil de almacenar las contraseñas es con el formato indicado, ya que anteponiendo la directiva “export” a cada una de ellas, permite que al ejecutar el fichero se creen las variables de entorno correspondientes. Para ejecutar dicho fichero solo hay que hacer:

```
1 controller#  . /root/passwordlist
```

7.4.1.2. Configuración de red

Configuración del nodo controller

El nodo controller necesita dos interfaces de red, una para la red provider y otra para la red management. Para ello hay que modificar el fichero /etc/network/interfaces:

```
1 controller#  vim /etc/network/interfaces
```

/etc/network/interfaces

```
1  # The primary network interface (Provider)
2  auto ens3
3  iface ens3 inet static
4  address 192.168.122.131
5  netmask 255.255.255.0
6  network 192.168.122.0
7  gateway 192.168.122.1
8  broadcast 192.168.122.255
9  dns-nameservers 192.168.122.1
10
11 # Network interface (Management)
12 auto ens8
13 iface ens8 inet static
14 address 10.8.8.131
15 netmask 255.255.255.0
16 network 10.8.8.0
17 broadcast 10.8.8.255
```

Hay que configurar la resolución de nombres, así no es necesario aprender las IPs de memoria. Para ello se modifica el fichero /etc/hosts.

```
1 controller#  vim /etc/hosts
```

/etc/hosts

```
1  127.0.1.1   controller
2  10.8.8.131 controller
3  10.8.8.63   compute
```

Configuración del nodo compute

El nodo compute también necesita dos interfaces de red, para ello hay que modificar el fichero /etc/network/interfaces

```
compute#  vim /etc/network/interfaces
```

/etc/network/interfaces

```
# The primary network interface (Provider)
auto ens3
iface ens3 inet static
address 192.168.122.63
netmask 255.255.255.0
network 192.168.122.0
gateway 192.168.122.1
broadcast 192.168.122.255
dns-nameservers 192.168.122.1

# Network interface (Management)
auto ens8
iface ens8 inet static
address 10.8.8.63
netmask 255.255.255.0
network 10.8.8.0
broadcast 10.8.8.255
```

Al igual que ocurría en el nodo controller hay que cambiar la resolución de nombres, modificando para ello el fichero /etc/hosts.

```
compute#       vim /etc/hosts
```

/etc/hosts

```
127.0.1.1   compute
10.8.8.131 controller
10.8.8.63  compute
```

Configuración común a ambos nodos

Antes de continuar instalando más servicios hay que reiniciar ambos nodos, esto permite que las interfaces de red se inicien correctamente con sus nuevos direccionamientos. Cuando se hayan encendido de nuevo, se puede probar la conectividad entre ambos nodos y con internet mediante los siguientes comandos:

```
$ ping openstack.org
$ ping compute
$ ping controller
```

Con esta prueba también se verifica que está funcionando correctamente la resolución de nombres, ya que cuando se hace ping a controller en realidad se está haciendo a la IP 10.8.8.131 y cuando se hace a compute es en realidad a la IP 10.8.8.63.

7.4.1.3. Network Time Protocol

NTP, es un protocolo usado para la sincronización de relojes de los sistemas informáticos, imprescindible para el correcto funcionamiento de OpenStack, ya que todos los nodos deben estar perfectamente sincronizados. La manera óptima de implementarlo es que el nodo controller haga referencia a un servidor NTP más preciso y el resto de los nodos referencien al nodo controller. Se ha elegido chrony para implementar NTP.

Configuración del nodo controller

Para instalar chrony en el nodo controller hay que ejecutar el siguiente comando:

```
1 controller# apt install chrony
```

Hay que referenciar un servidor NTP válido, es decir, que pertenezca a la misma región en la que vaya a estar ubicada la implementación de la plataforma. En el caso de España un posible servidor es “es.pool.ntp.org”.

```
1 controller# vim /etc/chrony/chrony.conf
```

/etc/chrony/chrony.conf

```
1 pool 2.debian.pool.ntp.org offline iburst
2 server es.pool.ntp.org iburst
3 allow 10.8.8.0/24
```

Hay que reiniciar el servicio para aplicar los cambios.

```
1 controller# service chrony restart
```

Configuración del nodo compute

En el nodo compute también hay que instalar chrony, el comando para instalarlo es el mismo usado anteriormente.

```
1 compute# apt install chrony
```

En este nodo en vez referenciar a un servidor NTP, hay configurarlo para que se sincronice con el nodo controller. Para ello hay que modificar el fichero /etc/chrony/chrony.conf:

```
1 compute# vim /etc/chrony/chrony.conf
```

/etc/chrony/chrony.conf

```
1 pool 2.debian.pool.ntp.org offline iburst
2 server controller iburst
```

De nuevo hay que reiniciar el servicio para aplicar los cambios.

```
compute# service chrony restart
```

Configuración común a ambos nodos

Antes de continuar instalando más componentes es recomendable comprobar el correcto funcionamiento del servicio. Para ello hay que ejecutar el siguiente comando en ambos nodos:

```
compute# chronyc sources
```

La Fig. 7.31 refleja como el nodo controller está sincronizado con el servidor NTP configurado anteriormente, siendo la IP mostrada la de dicho servidor.

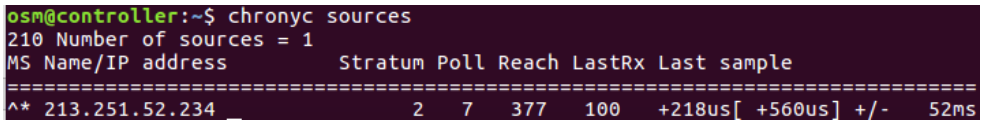

Fig. 7.31. Servicio NTP – Nodo controller

Por otro lado, la Fig. 7.32 muestra como el nodo compute está sincronizado con el nodo controller.

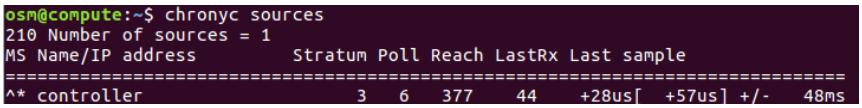

Fig. 7.32. Servicio NTP – Nodo compute

7.4.1.4. Repositorios de OpenStack

Para poder instalar posteriormente los servicios de OpenStack hace falta habilitar sus repositorios en todos los nodos de la arquitectura.

Configuración común a ambos nodos

Los siguientes comandos sirven para incluir los repositorios de Openstack en el listado de paquetes disponible de los nodos.

```
# apt install software-properties-common
# add-apt-repository cloud-archive:newton
```

Es conveniente que a la hora de instalar un paquete sea la versión más reciente. Para actualizar la lista de paquetes hay que ejecutar los siguientes comandos:

```
# apt update
# apt dist-upgrade
```

Para comprobar que los repositorios se añadieron correctamente se puede instalar el cliente de Openstack, componente que debe estar siempre presente independientemente de los servicios que se quieran implementar posteriormente.

```
# apt install python-openstackclient
```

7.4.1.5. Base de datos SQL

La mayoría de los servicios de OpenStack usan una base de datos SQL para almacenar información de manera persistente. Se ha elegido MariaDB para implementar la base de datos.

Configuración en el nodo controller

Para instalar la base de datos hay que ejecutar el siguiente comando:

```
1 controller#  apt install mariadb-server python-pymysql
```

Posteriormente hay que configurarla, que se consigue modificando el fichero /etc/mysql/mariadb.conf.d/99-openstack.cnf.

```
1 controller#  vim /etc/mysql/mariadb.conf.d/99-openstack.cnf
```

/etc/mysql/mariadb.conf.d/99-openstack.cnf

```
1  [mysqld]
2  bind-address = 10.8.8.131
3
4  default-storage-engine = innodb
5  innodb_file_per_table
6  max_connections = 4096
7  collation-server = utf8_general_ci
8  character-set-server = utf8
```

Para que los cambios introducidos se apliquen correctamente hay que reiniciar el servicio.

```
1 controller#  service mysql restart
```

Es recomendable hacer la implementación de la base de datos lo más segura posible, ya que va a contener toda la información de la plataforma. Una manera de hacerlo es asignarle una contraseña, para que solo los usuarios que la conozcan puedan acceder. Esto se puede conseguir mediante el siguiente comando:

```
1 controller#  mysql_secure_installation
```

Se debe proteger la base de datos mediante la contraseña DB_PASS, generada en el apartado 7.4.1.1.

7.4.1.6. Cola de mensajes

OpenStack utiliza una cola de mensajes para coordinar las operaciones y la información entre los diferentes servicios. Existen varias alternativas a la hora de implementarlo. Se ha optado por RabbitMQ ya que la mayoría de las distribuciones lo admiten.

Configuración del nodo controller.

Comando para instalar el servicio:

```
controller#  apt install rabbitmq-server
```

Hay que crear el usuario “openstack” con la contraseña RABBIT_PASS, generada en el apartado 7.4.1.1..

```
controller#  rabbitmqctl add_user openstack 0733cadb7ace0d5e7a40
```

Asignación de los permisos de lectura, escritura y ejecución al nuevo usuario:

```
controller#  rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

7.4.1.7. Memcached

El servicio de identidad de Openstack, más conocido como Keystone, usa Memcached para almacenar en caché los tokens de autenticación y así reducir el número de accesos a la base de datos.

Configuración nodo controller

Comando para instalar Memcached:

```
controller#  apt install memcached python-memcache
```

Es necesario configurar el servicio, para ello hay que modificar el fichero /etc/memcached.conf.

```
controller#  vim /etc/memcached.conf
```

/etc/memcached.conf

```
-l 127.0.0.1
-l 10.8.8.131
```

En último lugar hay que reiniciar el servicio para aplicar los cambios realizados.

```
controller#  service memcached restart
```

7.4.2. Keystone

En este apartado se recoge el procedimiento de instalación de Keystone, también conocido como el servicio de identidad, el cual se realiza únicamente en el nodo controller.

7.4.2.1. Instalación y configuración del nodo controller

- **Configuración de la base de datos:**

  En primer lugar, hay que crear en la base de datos la tabla necesaria para poder almacenar la información relativa al servicio de identidad.

```
1 controller#  mysql -u root -p$DB_PASS
2 mysql>       CREATE DATABASE keystone;
3 mysql>       GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED
               BY 'c1d3b017c5ceb579b8e2';
4 mysql>       GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY
               'c1d3b017c5ceb579b8e2';
5 mysql>       exit
```

- **Instalación del servicio de identidad**:

  Comando para instalar Keystone:

```
1 controller#  apt install keystone
```

  Configuración inicial del servicio.

```
1 controller#  vim /etc/keystone/keystone.conf
```

/etc/keystone/keystone.conf

```
1  [database]
2  connection = mysql+pymysql://keystone:c1d3b017c5ceb579b8e2@controller/keystone
3  connection = sqlite:////var/lib/keystone/keystone.db
4  [token]
5  provider = fernet
```

  Rellenar la base de datos de Keystone.

```
1 controller#  su -s /bin/sh -c "keystone-manage db_sync" keystone
```

  Inicialización de los repositorios.

```
1 controller#  keystone-manage fernet_setup --keystone-user keystone --keystone-group
               keystone
2 controller#  keystone-manage credential_setup --keystone-user keystone --keystone-group
               keystone
3 controller#  keystone-manage bootstrap --bootstrap-password 9cbcde54016694167ac9 --
               bootstrap-admin-url http://controller:35357/v3/ --bootstrap-internal-url
               http://controller:35357/v3/ --bootstrap-public-url http://controller:5000/v3/ --
               bootstrap-region-id RegionOne
```

- **Configuración del servidor HTTP Apache:**

```
1 controller#  vim /etc/apache2/apache2.conf
```

/etc/apache2/apache2.conf

```
1  ServerName controller
```

  Hay que reiniciar el servicio para que se apliquen todos los cambios realizados.

```
1 controller#  service apache2 restart
2 controller#  rm -f /var/lib/keystone/keystone.db
```

- **Configurar la cuenta administrativa “admin”**

  Hay que crear un script que al ejecutarlo proporcione los privilegios de la cuenta administrativa “admin”.

```
1 controller#  vi /root/admin-openrc
```

/root/admin-openrc

```
1  export OS_PROJECT_DOMAIN_NAME=Default
2  export OS_USER_DOMAIN_NAME=Default
3  export OS_PROJECT_NAME=admin
4  export OS_USERNAME=admin
5  export OS_PASSWORD=9cbcde54016694167ac9
6  export OS_AUTH_URL=http://controller:35357/v3
7  export OS_IDENTITY_API_VERSION=3
8  export OS_IMAGE_API_VERSION=2
```

  Para ejecutar el script hay que usar el siguiente comando:

```
1 controller#  . /root/admin-openrc
```

- **Configurar la cuenta administrativa “demo”**

  Hay que repetir el procedimiento realizado para la cuenta administrativa “admin”, solo que en este caso el script proporciona los privilegios de la cuenta administrativa “demo”:

```
1 controller#  vi /root/demo-openrc
```

/root/demo-openrc

```
1  export OS_PROJECT_DOMAIN_NAME=Default
2  export OS_USER_DOMAIN_NAME=Default
3  export OS_PROJECT_NAME=demo
4  export OS_USERNAME=demo
5  export OS_PASSWORD=11d649a6714d3380626a
6  export OS_AUTH_URL=http://controller:5000/v3
7  export OS_IDENTITY_API_VERSION=3
8  export OS_IMAGE_API_VERSION=2
```

  Para ejecutar el script, hay que utilizar el siguiente comando:

```
1 controller#  . /root/demo-openrc
```

7.4.2.2. Creación de domain, projects, users, and roles

Configuración del nodo controller

- **Creación de proyectos:**

  Creación del “Service Project”.

```
1 controller# openstack project create --domain default --description "Service Project" service
```

Creación del “Demo Project”

```
1 controller# openstack project create --domain default --description "Demo Project" demo
```

- **Creación de usuarios:**

  Creación del usuario “demo”

```
1 controller# openstack user create --domain default --password-prompt demo
```

- **Creación de roles:**

  Crear el rol “user”

```
1 controller# openstack role create user
```

  Añadir el rol “user” al proyecto “demo” y al usuario “user”.

```
1 controller# openstack role add --project demo --user demo user
```

7.4.2.3. Verificar funcionamiento

Para comprobar que el servicio funciona correctamente se pueden solicitar tokens con los diferentes usuarios ya creados: “admin” y “demo”.

- **Solicitar token como usuario “admin”:**

```
1 controller# openstack --os-auth-url http://controller:35357/v3 --os-project-domain-name
  Default --os-user-domain-name Default --os-project-name admin --os-username
  admin token issue
```

- **Solicitar un token como el usuario “demo”:**

```
1 controller# openstack --os-auth-url http://controller:5000/v3 --os-project-domain-name
  Default --os-user-domain-name Default --os-project-name demo --os-username
  demo token issue
```

7.4.3. Glance

En este apartado se recoge el procedimiento de instalación de Glance, también conocido como el servicio de imagen, el cual se realiza únicamente en el nodo controller.

7.4.3.1. Instalación y configuración del nodo controller

- **Configuración de la base de datos:**

  En primer lugar, hay que crear en la base de datos la tabla necesaria para poder almacenar la información relativa al servicio de imagen.

```
1 controller# mysql -u root -p$DB_PASS
2 mysql>      CREATE DATABASE glance;
3 mysql>      GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY
  'cda9bd2eecea3d47a670';
```

```
4 mysql>  GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY
          'cda9bd2eecea3d47a670';
5 mysql>  exit
```

- **Creación de las credenciales de Glance**

  Cargar las credenciales del usuario “admin” para obtener acceso.

```
1 controller# . admin-openrc
```

  Creación del usuario, rol y servicio, correspondientes a Glance.

```
1 controller# openstack user create --domain default --password-prompt glance
2 controller# openstack role add --project service --user glance admin
3 controller# openstack service create --name glance --description "OpenStack Image" image
```

  Creación de los API endpoints correspondientes al servicio de imagen.

```
1 controller# openstack endpoint create --region RegionOne image public http://controller:9292
2 controller# openstack endpoint create --region RegionOne image internal
              http://controller:9292
3 controller# openstack endpoint create --region RegionOne image admin http://controller:9292
```

- **Instalación y configuración de los componentes:**

  Comando para instalar Glance:

```
1 controller# apt install glance
```

  Configuración inicial del servicio.

```
1 controller# vim /etc/glance/glance-api.conf
```

/etc/glance/glance-api.conf

```
1  [database]
2  connection = mysql+pymysql://glance:cda9bd2eecea3d47a670@controller/glance
3  [keystone_authtoken]
4  auth_uri = http://controller:5000
5  auth_url = http://controller:35357
6  memcached_servers = controller:11211
7  auth_type = password
8  project_domain_name = Default
9  user_domain_name = Default
10 project_name = service
11 username = glance
12 password = c10218b2e490f5822f6d
13 [paste_deploy]
14 flavor = keystone
15 [glance_store]
```

```
16  stores = file,http
17  default_store = file
18  filesystem_store_datadir = /var/lib/glance/images/
```

```
1 controller#  vim /etc/glance/glance-registry.conf
```

/etc/glance/glance-registry.conf

```
1   [database]
2   connection = mysql+pymysql://glance:cda9bd2eecea3d47a670@controller/glance
3   [keystone_authtoken]
4   auth_uri = http://controller:5000
5   auth_url = http://controller:35357
6   memcached_servers = controller:11211
7   auth_type = password
8   project_domain_name = Default
9   user_domain_name = Default
10  project_name = service
11  username = glance
12  password = c10218b2e490f5822f6d
13  [paste_deploy]
14  flavor = keystone
```

Rellenar la base de datos de Glance.

```
1 controller#  su -s /bin/sh -c "glance-manage db_sync" glance
```

Reiniciar el servicio para aplicar los cambios realizados.

```
1 controller#  service glance-registry restart
2 controller#  service glance-api restart
```

7.4.3.2. Verificar funcionamiento

Para comprobar que el servicio funciona correctamente se debe crear una imagen. En este caso se ha elegido la imagen del sistema operativo CirrOS debido a los pocos recursos que necesita.

- **Creación de la imagen**

  Cargar las credenciales del usuario “admin” para obtener acceso.

  ```
  1 controller#  . /root/admin-openrc
  ```

  Descargar la imagen de CirrOS.

  ```
  1 controller#  wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
  ```

  Incorporar la imagen recién descargada a OpenStack, para que posteriormente se puedan crear instancias.

```
1 controller# openstack image create "cirros" --file cirros-0.3.4-x86_64-disk.img --disk-format
  qcow2 --container-format bare --public
```

- **Comprobación**

  Ver una lista de las imágenes disponibles para crear instancias, Fig. 7.33.

```
1 controller# openstack image list
```

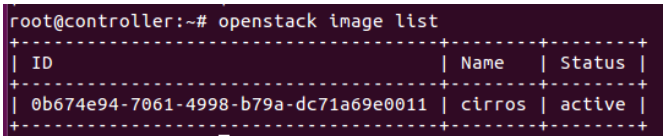

Fig. 7.33. Listado de imágenes

7.4.4. Nova

En este apartado se recoge el procedimiento de instalación de Nova, también conocido como el servicio de cómputo, el cual se realiza sobre los nodos controller y compute.

7.4.4.1. Instalación y configuración del nodo controller

- **Configuración de la base de datos:**

  En primer lugar, hay que crear en la base de datos las tablas necesarias para poder almacenar la información relativa al servicio de cómputo.

```
1 controller# mysql -u root -p$DB_PASS
2 mysql>      CREATE DATABASE nova_api;
3 mysql>      CREATE DATABASE nova;
4 mysql>      GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED
              BY 'd95dc71b40111b6fcf16';
5 mysql>      GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY
              'd95dc71b40111b6fcf16';
6 mysql>      GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY
              'd95dc71b40111b6fcf16';
7 mysql>      GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY
              'd95dc71b40111b6fcf16';
8 mysql>      exit
```

- **Creación de las credenciales de Nova:**

  Cargar las credenciales del usuario “admin” para obtener acceso.

```
1 controller# . admin-openrc
```

  Creación del usuario, rol y servicio, correspondiente a Nova:

```
1 controller# openstack user create --domain default --password-prompt nova
2 controller# openstack role add --project service --user nova admin
3 controller# openstack service create --name nova --description "OpenStack Compute"
              compute
```

Creación del API endpoints de Nova

```
1 controller# openstack endpoint create --region RegionOne compute public http://controller:8774/v2.1/%\(tenant_id\)s
2 controller# openstack endpoint create --region RegionOne compute internal http://controller:8774/v2.1/%\(tenant_id\)s
3 controller# openstack endpoint create --region RegionOne compute public http://controller:8774/v2.1/%\(tenant_id\)s openstack endpoint create --region RegionOne compute admin http://controller:8774/v2.1/%\(tenant_id\)s
```

- **Instalación y configuración de los componentes:**

Instalar los paquetes de Nova:

```
1 controller# apt install nova-api nova-conductor nova-consoleauth nova-novncproxy nova-scheduler
```

Configurar el servicio:

```
1 controller# vim /etc/nova/nova.conf
```

/etc/nova/nova.conf

```
1  [api_database]
2  connection = mysql+pymysql://nova:d95dc71b40111b6fcf16@controller/nova_api
3  [database]
4  connection = mysql+pymysql://nova:d95dc71b40111b6fcf16@controller/nova
5  [DEFAULT]
6  transport_url = rabbit://openstack:0733cadb7ace0d5e7a40@controller
7  auth_strategy = keystone
8  my_ip = 10.0.0.16
9  use_neutron = True
10 firewall_driver = nova.virt.firewall.NoopFirewallDriver
11 log-dir=/var/log/nova
12 [keystone_authtoken]
13 auth_uri = http://controller:5000
14 auth_url = http://controller:35357
15 memcached_servers = controller:11211
16 auth_type = password
17 project_domain_name = Default
18 user_domain_name = Default
19 project_name = service
20 username = nova
21 password = 8676dc35c60fc95b10a3
22 [vnc]
23 vncserver_listen = $my_ip
24 vncserver_proxyclient_address = $my_ip
25 [glance]
26 api_servers = http://controller:9292
```

```
27  [oslo_concurrency]
28  lock_path = /var/lib/nova/tmp
```

Rellenar la base de datos:

```
1  controller#       su -s /bin/sh -c "nova-manage api_db sync" nova
2  controller#  su -s /bin/sh -c "nova-manage db sync" nova
```

- **Finalizar la instalación:**

  Reiniciar los servicios para aplicar los cambios realizados:

```
1  controller#  service nova-api restart
2  controller#  service nova-consoleauth restart
3  controller#  service nova-scheduler restart
4  controller#  service nova-conductor restart
5  controller#  service nova-novncproxy restart
```

7.4.4.2. Instalación y configuración del nodo compute

- **Instalación de los componentes**

  Instalar Nova.

```
1  compute#  apt install nova-compute
```

  Configurar el servicio.

```
1  compute#  vim /etc/nova/nova.conf
```

/etc/nova/nova.conf

```
1   [DEFAULT]
2   transport_url = rabbit://openstack:0733cadb7ace0d5e7a40@controller
3   auth_strategy = keystone
4   my_ip = 10.0.0.229
5   use_neutron = True
6   firewall_driver = nova.virt.firewall.NoopFirewallDriver
7   log-dir=/var/log/nova
8   [keystone_authtoken]
9   auth_uri = http://controller:5000
10  auth_url = http://controller:35357
11  memcached_servers = controller:11211
12  auth_type = password
13  project_domain_name = Default
14  user_domain_name = Default
15  project_name = service
16  username = nova
17  password = 8676dc35c60fc95b10a3
18  [vnc]
```

```
19  enabled = True
20  vncserver_listen = 0.0.0.0
21  vncserver_proxyclient_address = $my_ip
22  novncproxy_base_url = http://controller:6080/vnc_auto.html
23  [glance]
24  api_servers = http://controller:9292
25  [oslo_concurrency]
26  lock_path = /var/lib/nova/tmp
```

- **Aceleración hardware:**

  Determinar si el nodo compute soporta la aceleración hardware para las máquinas virtuales.

```
1 compute#  egrep -c '(vmx|svm)' /proc/cpuinfo
```

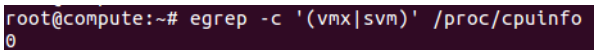

Fig. 7.34. Comprobación aceleración hardware

Si el comando devuelve 0, como se puede ver en la Fig. 7.34, significa que el equipo no soporta la aceleración hardware y hay que realizar la siguiente configuración adicional:

```
1 compute#  vim /etc/nova/nova-compute.conf
```

/etc/nova/nova-compute.conf

```
1  [libvirt]
2  virt_type=kvm
3  virt_type=qemu
```

- **Finalizar la instalación:**

  Reiniciar el servicio para aplicar los cambios.

```
1 controller#  service nova-compute restart
```

7.4.4.3. Verificar funcionamiento

Para comprobar que el servicio funciona correctamente se debe listar todos los componentes de Nova para ver si se han instalado correctamente.

- **Ver listado de los servicios:**

  Cargar las credenciales del usuario “admin” para obtener acceso.

```
1 controller#  . admin-openrc
```

  Ver la lista de servicios, Fig. 7.35.

```
1 controller#  openstack compute service list
```

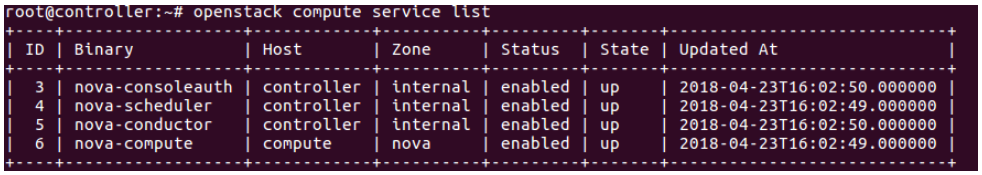

Fig. 7.35. Listado de servicios

7.4.5. Neutron

En este apartado se recoge el procedimiento de instalación de Neutron, también conocido como el servicio de red, el cual se realiza en los nodos controller y compute.

7.4.5.1. Instalación y configuración del nodo controller

- **Configuración de la base de datos:**

  En primer lugar, hay que crear en la base de datos la tabla necesaria para poder almacenar la información relativa al servicio Neutron.

```
1 controller#  mysql -u root -p~$DB_PASS
2 mysql>       CREATE DATABASE neutron;
3 mysql>       GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED
               BY '98317a93b286dbc55259';
4 mysql>       GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%'  IDENTIFIED BY
               '98317a93b286dbc55259';
5 mysql>       exit
```

- **Creación de las credenciales de Neutron:**

  Cargar las credenciales del usuario “admin” para obtener acceso.

```
1 controller#  . /root/admin-openrc
```

  Creación del usuario, rol y servicio, correspondientes a Neutron.

```
1 controller#  openstack user create --domain default --password-prompt neutron
2 controller#  openstack role add --project service --user neutron admin
3 controller#  openstack service create --name neutron --description "OpenStack Networking"
               network
```

  Creación del API endpoints del servicio de red.

```
1 controller#  openstack endpoint create --region RegionOne network public
               http://controller:9696
2 controller#  openstack endpoint create --region RegionOne network internal
               http://controller:9696
3 controller#  openstack endpoint create --region RegionOne network admin
               http://controller:9696
```

- **Configuración de los paquetes de red.**

  Instalar Neutron.

```
1 controller# apt install neutron-server neutron-plugin-ml2 neutron-linuxbridge-agent
              neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent
```

Hay que editar varios ficheros de configuración.

```
1 controller# vim /etc/neutron/neutron.conf
```

/etc/neutron/neutron.conf

```
1  [database]
2  connection = mysql+pymysql://neutron:98317a93b286dbc55259@controller/neutron
3  connection = sqlite:////var/lib/neutron/neutron.sqlite
4  [DEFAULT]
5  core_plugin = ml2
6  service_plugins = router
7  allow_overlapping_ips = True
8  transport_url = rabbit://openstack:0733cadb7ace0d5e7a40@controller
9  auth_strategy = keystone
10 notify_nova_on_port_status_changes = True
11 notify_nova_on_port_data_changes = True
12 [nova]
13 auth_url = http://controller:35357
14 auth_type = password
15 project_domain_name = Default
16 user_domain_name = Default
17 region_name = RegionOne
18 project_name = service
19 username = nova
20 password = 8676dc35c60fc95b10a3
```

```
1 controller# vim /etc/neutron/plugins/ml2/ml2_conf.ini
```

/etc/neutron/plugins/ml2/ml2_conf.ini

```
1  [ml2]
2  type_drivers = flat,vlan,vxlan
3  tenant_network_types = vxlan
4  mechanism_drivers = linuxbridge,l2population
5  extension_drivers = port_security
6  [ml2_type_flat]
7  flat_networks = provider
8  [ml2_type_vxlan]
9  vni_ranges = 1:1000
10 [securitygroup]
11 enable_ipset = True
```

```
1 controller# vim /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

/etc/neutron/plugins/ml2/linuxbridge_agent.ini

```
1  [linux_bridge]
2  physical_interface_mappings = provider:ens3
3  [vxlan]
4  enable_vxlan = True
5  local_ip = 10.0.0.16
6  l2_population = True
7  [securitygroup]
8  enable_security_group = True
9  firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

```
1 controller#  vim /etc/neutron/l3_agent.ini
```

/etc/neutron/l3_agent.ini

```
1  [DEFAULT]
2  interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
```

```
1 controller#  vim /etc/neutron/dhcp_agent.ini
```

/etc/neutron/l3_agent.ini

```
1  [DEFAULT]
2  interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
3  dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
4  enable_isolated_metadata = True
```

```
1 controller#  vim /etc/neutron/metadata_agent.ini
```

/etc/neutron/metadata_agent.ini

```
1  [DEFAULT]
2  nova_metadata_ip = controller
3  metadata_proxy_shared_secret = 0abb3f64fa1034271455
```

- **Configuración de Nova**

  Hay que configurar el servicio Nova para que interactúe correctamente con Neutron.

```
1 controller#  vim /etc/nova/nova.conf
```

/etc/nova/nova.conf

```
1  [neutron]
2  url = http://controller:9696
3  auth_url = http://controller:35357
4  auth_type = password
5  project_domain_name = Default
6  user_domain_name = Default
7  region_name = RegionOne
```

```
project_name = service
username = neutron
password = a23423041cf257196070
service_metadata_proxy = True
metadata_proxy_shared_secret = 0abb3f64fa1034271455
```

- **Finalizar la instalación**

  Rellenar la base de datos.

```
controller# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

  Reiniciar los servicios para aplicar los cambios realizados.

```
controller# service nova-api restart
controller# service neutron-server restart
controller# service neutron-linuxbridge-agent restart
controller# service neutron-dhcp-agent restart
controller# service neutron-metadata-agent restart
controller# service neutron-l3-agent restart
```

7.4.5.2. Instalación y configuración del nodo compute

- **Instalación de los componentes.**

  Instalación de Neutron.

```
compute# apt install neutron-linuxbridge-agent
```

  Configuración del servicio.

```
compute# vim /etc/neutron/neutron.conf
```

/etc/neutron/neutron.conf

```
[database]
connection = sqlite:////var/lib/neutron/neutron.sqlite
[DEFAULT]
transport_url = rabbit://openstack:0733cadb7ace0d5e7a40@controller
auth_strategy = keystone
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
```

```
14  username = neutron
15  password = a23423041cf257196070
```

- **Configuración de las opciones de red**

  Hay que editar varios ficheros de configuración.

```
1 compute#  vim /etc/neutron/plugins/ml2/linuxbridge_agent.ini
```

/etc/neutron/plugins/ml2/linuxbridge_agent.ini

```
1  [linux_bridge]
2  physical_interface_mappings = provider:ens3
3  [vxlan]
4  enable_vxlan = True
5  local_ip = 10.0.0.229
6  l2_population = True
7  [securitygroup]
8  enable_security_group = True
9  firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

```
1 compute#  vim /etc/nova/nova.conf
```

/etc/nova/nova.conf

```
1  [neutron]
2  url = http://controller:9696
3  auth_url = http://controller:35357
4  auth_type = password
5  project_domain_name = Default
6  user_domain_name = Default
7  region_name = RegionOne
8  project_name = service
9  username = neutron
10 password = a23423041cf257196070
```

- **Finalizar la instalación**

  Reiniciar los servicios para que se apliquen los cambios realizados.

```
1 compute#  service nova-compute restart
2 compute#  service neutron-linuxbridge-agent restart
```

7.4.5.3. Verificar funcionamiento

- **Obtención lista de las extensiones**

  Cargar las credenciales del usuario “admin” para obtener acceso.

```
1 controller#  . admin-openrc
```

- **Obtener una lista de las extensiones de Neutron, Fig. 7.36 y Fig. 7.37.**

```
1 controller# neutron ext-list
```

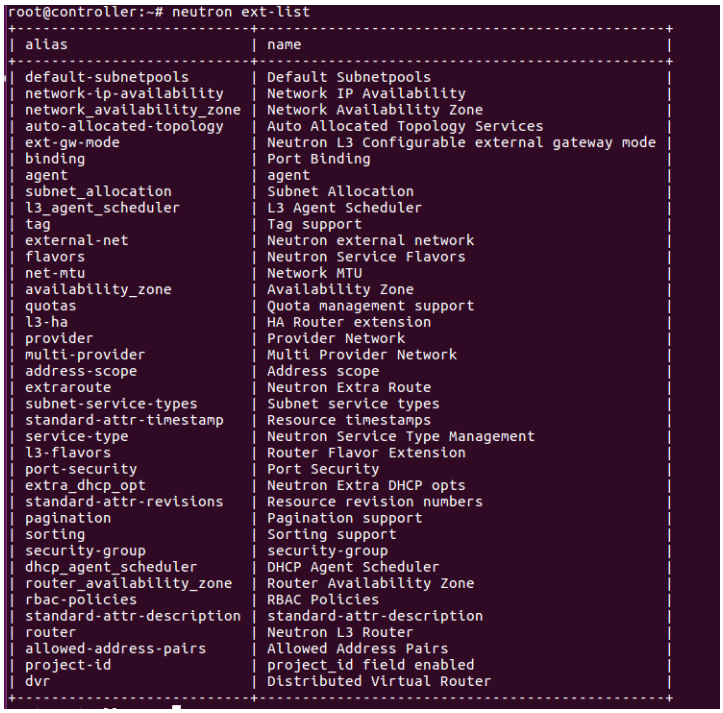

Fig. 7.36. Lista de extensiones

```
1 controller# openstack network agent list
```

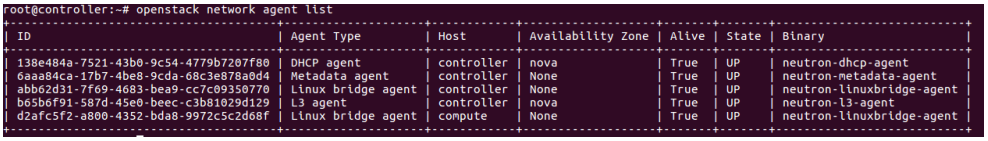

Fig. 7.37. Lista de agentes

7.4.6. Horizon

En este apartado se recoge el procedimiento de instalación de Horizon, el dashboard de Openstack, el cual se realiza únicamente en el nodo controller.

7.4.6.1. Instalación y configuración del nodo controller

- **Instalar los componentes**

  Instalar dashboard.

```
1 controller# apt install openstack-dashboard
```

  Configuración.

```
1 controller# vim /etc/openstack-dashboard/local_settings.py
```

/etc/openstack-dashboard/local_settings.py

```
OPENSTACK_HOST = "controller"
ALLOWED_HOSTS = ['*', ]
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': 'controller:11211',
    }
}
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
TIME_ZONE = "Europe/Madrid"
```

- **Finalizar instalación**

  Reiniciar el servidor web para que se apliquen los cambios realizados

```
controller#  service apache2 reload
```

7.4.6.2. Verificar operación

En primer lugar, hay que crear una ruta en el equipo host para poder acceder al dashboard, ya que pertenecen a diferentes subredes.

```
host#  sudo route add 10.8.8.131 gw 192.168.122.131
```

Para acceder hay que introducir la siguiente dirección en un navegador web: http://10.8.8.131/horizon. Si todo funciona correctamente aparece la página de acceso a la Openstack, mostrada en la Fig. 7.38.

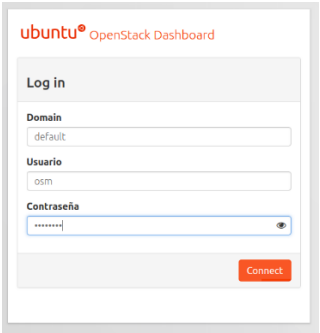

Fig. 7.38. Página de acceso a OpenStack

Glosario

La lista de todos los acrónimos utilizados en este estudio es la siguiente:

CAPEX Capital Expenditure

CP Connection Point

CPU Central Processing Unit

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

ETSI European Telecommunication Standards Institute

GRUB Grand Unifier Bootloader

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol

HW Hardware

ICMP Internet Control Message Protocol

ISG Industry Specification Group

LXC Linux Containers

LPI Ley de Propiedad Intelectual

LSSI Ley de Servicios de la Sociedad de la Información

LVM Logical Volume Manager

MANO Management and Orchestration

MON Monitoring

N2VC Network Service to VNF Communication

NBI Northbound Interface

NFV Network Function Virtualization

NFVI Network Functions Virtualization Infrastructure

NFVO NFV Orchestrator

NS Network Service

NSD Network Service Descriptor

OPEX Operational Expenditure

OS Operating System

OSM Open Source MANO

PM Policy Manager

RAM	Random Access Memory
RO	Resource Orchestrator
SDN	Software Defined Network
SO	Service Orchestrator
SQL	Structured Query Language
SSH	Secure Shell
SW	Software
TFG	Trabajo Fin de Grado
UI	User Interface
URL	Uniform Resource Locator
VCA	VNF Configuration and Abstraction
VDU	Virtual Deployment Unit
VIM	Virtual Infrastructure Manager
VL	Virtual Link
VLC	VideoLAN
VM	Virtual Machine
VNF	Virtualized Network Function
VNFD	Virtual Network Virtualization Descriptor
VNFM	VNF Manager

Bibliografía

[1] ISG NFV White Paper, «Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action.,» 24 Octubre 2012. [En línea]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf. [Último acceso: 03 Septiembre 2018].

[2] Publications Office of the European Union, «Implications of the emerging technologies Software-Defined Networking and Network Function Virtualisation on the future Telecommunications Landscape,» 2016. [En línea]. Available: http://ec.europa.eu/newsroom/dae/document.cfm?doc_id=44565. [Último acceso: 03 Septiembre 2018].

[3] Jefatura del Estado, «Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.,» 12 10 2002. [En línea]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758. [Último acceso: 03 Septiembre 2018].

[4] ETSI, «Network Functions Virtualisation,» [En línea]. Available: https://www.etsi.org/technologies-clusters/technologies/689-network-functions-virtualisation. [Último acceso: 03 Septiembre 2018].

[5] Ministerio de Educación, «Real Decreto 1791/2010, de 30 de diciembre, por el que se aprueba el Estatuto del Estudiante Universitario.,» 31 12 2010. [En línea]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2010-20147. [Último acceso: 03 Septiembre 2018].

[6] Ministerio de Cultura, «Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia.,» 22 04 1996. [En línea]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930. [Último acceso: 03 Septiembre 2018].

[7] ISG NFV White Paper 2, «Network Functions Virtualisation: Network Operator Perspectives on Industry Progress,» 17 Octubre 2013. [En línea]. Available: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf. [Último acceso: 03 Septiembre 2018].

[8] SDxCentral, «NFV Infrastructure,» [En línea]. Available: https://www.sdxcentral.com/wp-content/uploads/2017/02/nfv-infrastructure-and-vim-nfv-framewrok.png. [Último acceso: 03 Septiembre 2018].

[9] IETF, «ETSI NFV Management and Orchestration - An Overview,» [En línea]. Available: https://www.ietf.org/proceedings/88/slides/slides-88-opsawg-6.pdf. [Último acceso: 03 Septiembre 2018].

[10] IEEE.org, «OpenSource MANO,» [En línea]. Available: https://sdn.ieee.org/images/files/newsletter/201607-odini-figure1.jpg. [Último acceso: 03 Septiembre 2018].

[11] Open Source MANO, «Release 0 Data Model Details,» [En línea]. Available: https://osm.etsi.org/wikipub/images/thumb/0/03/Vnf_internal.png/600px-Vnf_internal.png. [Último acceso: 03 Septiembre 2018].

[12] ETSI, «Open Source MANO,» [En línea]. Available: https://www.etsi.org/technologies-clusters/technologies/nfv/open-source-mano. [Último acceso: 03 Septiembre 2018].

[13] Open Source MANO, «OSM Topogy,» [En línea]. Available: https://osm.etsi.org/wikipub/images/thumb/b/b3/Osmtopology.png/900px-Osmtopology.png. [Último acceso: 03 Septiembre 2018].

[14] Open Source MANO, «OSM RELEASE FOUR TECHNICAL OVERVIEW,» Mayo 2018. [En línea]. Available: https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseFOUR-FINAL.pdf. [Último acceso: 03 Septiembre 2018].

[15] Red Hat, «CONTENEDORES,» [En línea]. Available: https://www.redhat.com/cms/managed-files/virtualization-vs-containers.png. [Último acceso: 03 Septiembre 2018].

[16] Telefónica, «OpenMANO,» [En línea]. Available: http://www.tid.es/es/innovacion-de-largo-plazo/network-innovation/telefonica-nfv-reference-lab/openmano. [Último acceso: 03 Septiembre 2018].

[17] VMware, «vCloud Director,» [En línea]. Available: https://www.vmware.com/products/vcloud-director.html. [Último acceso: 03 Septiembre 2018].

[18] Amazon Web Services, «Cloud Computing con Amazon Web Services,» [En línea]. Available: https://aws.amazon.com/es/what-is-aws/. [Último acceso: 03 Septiembre 2018].

[19] OpenStack, «What is OpenStack?,» [En línea]. Available: https://www.openstack.org/software/. [Último acceso: 03 Septiembre 2018].

[20] Grafana, «The analytics platform for all your metrics,» [En línea]. Available: https://grafana.com/grafana. [Último acceso: 03 Septiembre 2018].

[21] Open Source MANO, «OSM Metrics Architecture,» [En línea]. Available: https://osm.etsi.org/wikipub/images/thumb/b/ba/OSM_Metrics_Architecture_4.png/1200px-OSM_Metrics_Architecture_4.png. [Último acceso: 03 Septiembre 2018].

[22] Elastic, «Your Window into the Elastic Stack,» [En línea]. Available: https://www.elastic.co/products/kibana. [Último acceso: 03 Septiembre 2018].

[23] Superuser, «OpenStack's design evolution,» [En línea]. Available: http://superuser.openstack.org/wp-content/uploads/2016/10/Screenshot-154-793x603.png. [Último acceso: 03 Septiembre 2018].

[24] OpenStack, «Overview,» [En línea]. Available: https://docs.openstack.org/newton/install-guide-ubuntu/overview.html. [Último acceso: 03 Septiembre 2018].

[25] Virtualiza desde zero, «¿Qué es Openstack y por qué deberías saber de su existencia?,» [En línea]. Available: https://virtualizadesdezero.com/wp-content/uploads/2017/03/openstack-conceptual-arch-folsom-compressor.png. [Último acceso: 03 Septiembre 2018].

[26] OpenStack, «Keystone, the OpenStack Identity Service,» [En línea]. Available: https://docs.openstack.org/keystone/latest/. [Último acceso: 03 Septiembre 2018].

[27] OpenStack, «Welcome to Glance's documentation!,» [En línea]. Available: https://docs.openstack.org/glance/latest/. [Último acceso: 03 Septiembre 2018].

[28] OpenStack, «OpenStack Compute (nova),» [En línea]. Available: OpenStack Compute (nova). [Último acceso: 03 Septiembre 2018].

[29] OpenStack, «Welcome to Neutron's documentation!,» [En línea]. Available: https://docs.openstack.org/neutron/latest/. [Último acceso: 03 Septiembre 2018].

[30] OpenStack, «Horizon: The OpenStack Dashboard Project,» [En línea]. Available: https://docs.openstack.org/horizon/latest/. [Último acceso: 03 Septiembre 2018].

[31] Xataka, «Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas,» [En línea]. Available: https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas. [Último acceso: 03 Septiembre 2018].

[32] Red Hat, «¿Qué es KVM?,» [En línea]. Available: https://www.redhat.com/es/topics/virtualization/what-is-KVM. [Último acceso: 03 Septiembre 2018].

[33] Open Source MANO, «OSM Release FOUR,» [En línea]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Release_FOUR. [Último acceso: 03 Septiembre 2018].

[34] Open Source MANO, «OSM Release THREE,» [En línea]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Release_THREE. [Último acceso: 03 Septiembre 2018].

[35] OpenStack, «DevStack,» [En línea]. Available: https://docs.openstack.org/devstack/latest/. [Último acceso: 03 Septiembre 2018].

[36] OpenStack, «Network Layout,» [En línea]. Available: https://docs.openstack.org/newton/install-guide-ubuntu/_images/networklayout.png. [Último acceso: 03 Septiembre 2018].

[37] Open Source MANO, «Openstack configuration (Release FOUR),» [En línea]. Available: https://osm.etsi.org/wikipub/index.php/Openstack_configuration_(Release_FOUR). [Último acceso: 03 Septiembre 2018].

[38] Open Source MANO, «Cirros 2VNF NS,» [En línea]. Available: https://osm.etsi.org/wikipub/images/thumb/8/86/Cirros_2vnf_ns.png/750px-Cirros_2vnf_ns.png. [Último acceso: 03 Septiembre 2018].

[39] VideoLAN Wiki, «Documentation:Command line,» [En línea]. Available: https://wiki.videolan.org/Documentation:Command_line/. [Último acceso: 03 Septiembre 2018].

[40] GitHub, «Ampache Installation,» [En línea]. Available: https://github.com/ampache/ampache/wiki/Installation. [Último acceso: 03 Septiembre 2018].

[41] GitHub, «Ampache Catalog,» [En línea]. Available: https://github.com/ampache/ampache/wiki/Catalog. [Último acceso: 03 Septiembre 2018].

[42] Prácticas de los fundamentos Físicos en las ingenierías, «Las medidas experimentales y sus errores,» [En línea]. Available: http://umh1209.edu.umh.es/wp-content/uploads/sites/801/2015/01/Medidas-experimentales-y-errores.pdf. [Último acceso: 03 Septiembre 2018].

[43] Open Source MANO, «OSM Performance Management,» [En línea]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Performance_Management. [Último acceso: 03 Septiembre 2018].

[44] Open Source MANO, «Grafana Example,» [En línea]. Available: https://osm.etsi.org/wikipub/images/thumb/b/b3/OSM_Grafana_Example_4.png/1200px-OSM_Grafana_Example_4.png. [Último acceso: 03 Septiembre 2018].

[45] Open Source MANO, «OSM Fault Management,» [En línea]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Fault_Management. [Último acceso: 03 Septiembre 2018].

[46] Open Source MANO, «OSM Kibana,» [En línea]. Available: https://osm.etsi.org/wikipub/images/thumb/b/b0/OSM_Kibana.png/1200px-OSM_Kibana.png. [Último acceso: 03 Septiembre 2018].

[47] M. Peuster, H. Karl y S. v. Rossem, «MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments,» de *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Palo Alto, CA, USA, 2016.

[48] Open Source MANO, «VIM emulator,» [En línea]. Available: https://osm.etsi.org/wikipub/index.php/VIM_emulator. [Último acceso: 03 Septiembre 2018].

[49] Colegio Oficial de Ingenieros de Telecomunicación , «Perfil Socio Profesional del Ingeniero de Telecomunicaciones,» [En línea]. Available: https://www.coit.es/informes/informe-socioprofesional-coitaeit-mapa-del-titulado-de-ingenieria-de-telecomunicacion/mapa. [Último acceso: 17 Septiembre 2018].

[50] Linux Config, «Install And Set Up KVM,» [En línea]. Available: https://linuxconfig.org/install-and-set-up-kvm-on-ubuntu-18-04-bionic-beaver-linux. [Último acceso: 03 Septiembre 2018].

[51] CentOS, «LVM (Logical Volume Manager),» [En línea]. Available: https://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-lvm.html. [Último acceso: 03 Septiembre 2018].

[52] GNU, «GNU GRUB,» [En línea]. Available: https://www.gnu.org/software/grub/. [Último acceso: 03 Septiembre 2018].

[53] OpenStack, «OpenStack Installation Tutorial for Ubuntu,» [En línea]. Available: https://docs.openstack.org/newton/install-guide-ubuntu/. [Último acceso: 03 Septiembre 2018].